Vol.2009-CG-136 No.3
2009/8/20

ハードウェア回路実装のための半透明物体実時間描画アルゴリズム

山　口　悟　志[†1]

本論文は，半透明物体の実時間描画を目標とするハードウェア回路実装可能なアルゴリズムを提案するものである．表面下散乱の基本モデルには BSSRDF を使用した．テクスチャマップとしてサンプルデータを保存する代わりに，オブジェクト頂点ごとの一次元サンプリングリストとすることで必要なメモリ容量を減らし，また，BSSRDF の表面下散乱方程式に RAM テーブルを使用することで高速化を図り，回路実装時に実時間性を得られるものとした．

A Real-Time Hardware-Based Rendering Algorithm for Translucent Materials

Satoshi Yamaguchi [†1]

This paper describes the hardware implementable real-time rendering algorithm for translucent materials. The algorithm uses the basis of BSSRDF model in subsurface scattering term. Because of adopting some vertices as sampling points instead of generating huge texture maps, memory cost has been reduced. The paper also describes a technique of RAM table for BSSRDF diffuse reflectance function.

1. はじめに

半透明物体は我々の身の回りに広く存在し，皮膚や牛乳，大理石などの柔らかな透過光の表現は，フォトリアリスティックな映像表現の中では重要な要素の一つである．しかしながら，半透明物体を通過する光の経路は複雑なため，それら全てを正確に表現するには反射や屈折，散乱などについて膨大な計算量が必要となる．

Jensen らによる BSSRDF の双極子近似[1)] は，物体内の複雑な散乱光を仮想光源によって表現することで表面下散乱の近似に有効とされており，さらにその高速化手法として，物体全体の放射照度サンプルを得た後に拡散光を算出するという段階的手法[2)] が提案されている．これらは後の多くの研究の基礎となっている．

また，グラフィックスアクセラレータを利用した研究では，予め計算しておいた情報をテクスチャとして格納しておき，本レンダリングの際に使用する手法がある．例えば Dachsbacher らによるトランスルーセントシャドウマップ法[3)] は，シャドウマップ法を拡張して，光源から見た物体表面の距離や法線・放射照度といった表面下散乱光の計算に必要な情報をいくつかのマップに保存して半透明性を表現している．また，接平面から重点サンプリングを行うことで皮膚などの局所的な表面下散乱を高速に描画する Mertens らの手法[4)] や，Lensch らによるテクスチャアトラスを使用し全体の拡散光と局所的な拡散光を計算する手法[5)]，それを発展させセラジオシティ法を使用し 3 パスで描画する Carr らの手法[6)] などがあり，それぞれ高速な描画を実現している．

上記研究がある中で，我々はこれらのような既存のグラフィックスアクセラレータの利用ではなく，新たなハードウェアへの実装が可能なアルゴリズムの提案を目的としている．

新たなハードウェアとする理由は，前述の通り半透明の質感はあらゆる物体の表現に必要とされている上，近似により計算量が減っているとはいえ依然として複雑な計算により処理量が膨大となっているため，リアルタイムで描画するためには専用のハードウェアによる集中処理が必要であると考えたためである．

本論文が提案するアルゴリズムは，以下の特徴がある．

- 一次元リスト構造を用いた頂点サンプリング
  オブジェクトの頂点をサンプリング点とすることでメイン処理中においてサンプリング点の位置などを決定する労力を減らし，また二次元のマップを使用しないためメモリ領域の占有率を減らすことが出来る．加えて，頂点単位のサンプリングはサンプリング数が光源数の増加に殆ど影響を受けない利点もある．
- BSSRDF 拡散反射率関数の RAM テーブル化
  多数の演算器が必要で処理負荷の高い表面下散乱の拡散成分を得るために RAM テーブルを用いることで，ハードウェアコストの削減や高速化を可能としている．

これらの特徴を持つアルゴリズムであるため回路化が可能であり，半透明物体のリアルタ

†1 法政大学 情報科学研究科
Graduate School of Computer and Information Sciences, Hosei University

Vol.2009-CG-136 No.3
2009/8/20

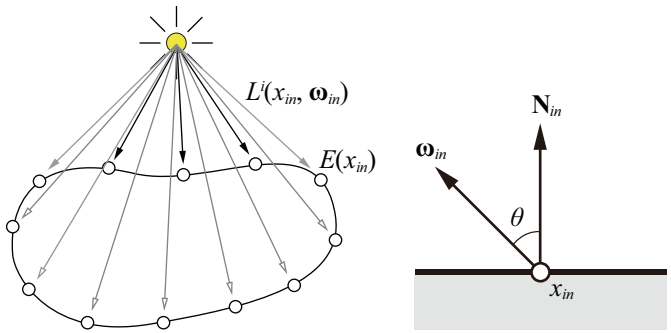

図 1 (左) 放射照度サンプリングのモデル (右) サンプリング点におけるベクトル
Fig. 1 (Left) A model of irradiance sampling. (Right) Vectors at sampling point.

イム描画を実現できる．本論文ではアルゴリズムに使用したモデルや処理フローについて説明する．

2. シェーディングアルゴリズム

2.1 表面下散乱方程式

本アルゴリズムは，BSSRDF の段階的手法[2] をベースに以下の式を用いて 2 パス方式にて描画する．サンプリングに関しては次節で詳細を述べることにし，ここでは式についてサンプリングモデルに関連させて説明する．

$$L^e(x_{out}, \omega_{out}) = \frac{1}{\pi} Ft(\eta, \omega_{out}) B(x_{out}) \tag{1}$$

$$B(x_{out}) = \int_{\mathsf{S}} E(x_{in}) R_d(r) dx_{in} \tag{2}$$

$$E(x_{in}) = \int_{\Omega^+(x_{in})} L^i(x_{in}, \omega_{in}) Ft(\eta, \omega_{in}) |\omega_{in} \cdot N_{in}| d\omega_{in} \tag{3}$$

まず第 1 パスでは，図 1（左）にあるように物体表面における放射照度 $E(x_{in})$ を算出し，RAM にサンプリングデータとして格納する．

x_{in} は各サンプリング点の位置，ω_{in} は x_{in} における光源の向き，N_{in} は x_{in} における法線を示す．η は物体の相対屈折率であり，フレネル項 $Ft(\eta, \omega_{in})$ で使用する．フレネル項には Schlick が提案した近似法[7] を使用し，N_{in} と ω_{in} の内積を $\cos\theta$ として計算する．

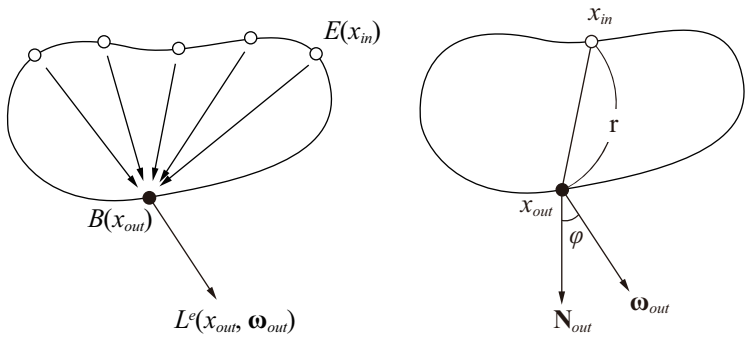

図 2 (左) 描画点における放射照度サンプルの積算モデル (右) 描画点におけるベクトル
Fig. 2 (Left) A model of irradiance sample accumulation at rendering point. (Right) Vectors at rendering point.

次に第 2 パスでは，各サンプリング点の放射照度 $E(x_{in})$ を，$x_{in}, x_{out} \in \mathsf{S}$ における BSSRDF 拡散反射率関数 $R_d(r)$ とフレネル項 $Ft(\eta, \omega_{out})$ でスケールする．r は入射点と出射点の距離 $|x_{out} - x_{in}|$ とする．フレネル項は，図 2 のように視線 ω_{out} と，x_{out} における法線 N_{out} の内積 $\cos\phi$ で求められる．

2.2 サンプリングアルゴリズム

これまでのいくつかの研究は，サンプリングにおいてオブジェクト表面上のデータを得るためにそのオブジェクト全体を事前に描画するなどの処理が必要であった[3][4][5][6]．

しかし，我々のアルゴリズムは，オブジェクトの頂点をサンプリング点とすることで全体を描画する必要はなくなり，メモリの使用量や処理負荷の削減を可能とした．

以下で図 3 を参考に，第 1 パスで行われるサンプリングプロセスについて説明する．

- サンプリング ターゲット リスト (STL)
  サンプリング前の準備として，サンプリング点になる各頂点の座標 x_{in} とその点における法線 N_{in} を全て，サンプリングターゲットリストと名付けられた RAM にロードする．
- アダプティブ サンプリング リスト (ASL)
  次に，STL のデータを使用して，各サンプリング点の放射照度 $E(x_{in})$ を算出し，アダプティブサンプリングリスト (ASL) と名付けられた RAM に格納していく．この時，放射照度がある閾値を上回ったサンプリング点のみを有効なサンプルとして ASL に格

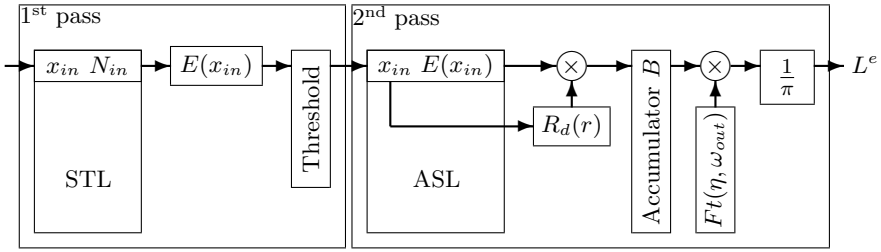

図 3 表面下散乱光取得の全体フロー
Fig. 3 All process flow to get subsurface scattering energy.

納する．これにより影響の僅かなサンプルを削減し，処理量を減らすことが出来る．

STL にロードするサンプリング点はどのようなものでも良いが，基本的には位置や法線などが全体的に均一になるように設定し，光源位置などを変化させて再サンプリングした場合にも適切な結果が得られるようにすることを推奨する．

ただし，意図的にサンプリング点を偏らせることで特殊な効果を狙うことも出来る．

また，オブジェクトの頂点数が少ない場合は，表面上に仮想的にサンプリング点を定義して補う必要がある．

上記手順で得られた ASL の最適化済みサンプリングデータは，図 3 の通り第 2 パスの拡散反射率計算で使用されるが，そのアクセスに特別な手順はなく，先頭から順に全てのデータを使用していく．これによりデータへのアクセス速度を効率化した．

3. 実　　装

3.1 表面下散乱光取得の全体フロー

第 1 パスにおいて STL にサンプリングターゲットをロードし放射照度 E を計算，ASL に格納する部分はサンプリングアルゴリズムとして 2.2 節で述べたとおりなので，ここでは図 3 をもとに第 2 パスについて説明する．

まず，描画点 x_{out} と ASL に格納された x_{in} の差分を距離 r とし，BSSRDF 拡散反射率関数テーブルのアドレスとする．そのアドレスを元にテーブルから R_d を得て，x_{in} とともに ASL にある E と乗算し，積算変数 B に加算する．この処理を ASL 内のデータ全てにおいて行う．

全てのデータにおける ER_d を B に積算した後は，B をフレネル項 $Ft(\eta, x_{out})$ と $1/\pi$ でスケールすることで L^e を得て，これを表面下散乱光とする．

表 1 シミュレーション結果
Table 1 Simulation result.

モデル名	頂点数	ポリゴン数	サンプリング頂点数		描画時間
			STL	ASL	(sec.)
Bunny（表面下散乱なし）	35,947	69,451	-	-	3.497
Bunny（R_d テーブルなし）			1,449	587	47.525
Bunny					6.775
Bunny（2 光源）				1,112	10.427
Bunny（3 光源）				1,244	12.330
Buddha（表面下散乱なし）	118,214	239,722	-	-	6.198
Buddha			1,890	798	11.787
Lucy（表面下散乱なし）	262,909	525,814	-	-	11.927
Lucy			1,413	653	14.674

3.2 BSSRDF 拡散反射率関数テーブル

BSSRDF 拡散反射率関数 $R_d(r)$ は入射点と出射点の距離 r による一次元関数であるため，RAM テーブル化することができ，それにより関数内の複雑な計算を省くことが出来る．

このテーブルには，オブジェクトの吸収率や散乱率などを定数とし，距離を変数として算出した値を納めていく．

3.3 ソフトウェアシミュレーション

本アルゴリズムによるソフトウェアシミュレーションを行い，その妥当性を検証した．

シミュレーション環境は Intel Core 2 Quad Q9650 (3.0GHz) プロセッサ，4GB メモリを使用し，ビデオカード等のアクセラレーションは用いていない．使用した言語は Java で，プログラム中にマルチスレッド処理は入っていない．作成したイメージサイズは 512×512 で，BSSRDF 拡散反射率関数 $R_d(r)$ のテーブルサイズは 1024 となっている．STL と ASL の最大要素数は 2048 とした．

STL にロードするサンプリング点群の生成については，以下の方法について試行した．

- 全頂点から抽出（最大要素数までリニア/ランダムに）
- 分割されたグループから抽出（最大要素数以下でリニア/ランダムに）
  - 頂点位置グループから抽出 (各軸 5/10/15/18/20 分割)
  - 法線グループから抽出 (角度によって 8/32 分割)
  - 頂点位置と法線の組み合わせを考慮したグループから抽出

上記サンプリング点群の生成や，シミュレーションの結果について，次章で考察する．

図 4 BSSRDF 拡散反射率関数テーブルを使用した場合の違い
(左) テーブルを使用した画像 (中) テーブルを使用しなかった画像 (右) (左) と (中) の差の絶対値
Fig. 4 Difference of using/not using BSSRDF Diffusion Table.
(Left) Using. (Right) Not using. (Bottom) Difference of (Left) and (Right).

4. 考　　察

4.1 実 験 結 果

4.1.1 サンプリング点群生成

レンダリング前準備として行うサンプリング点群生成について，3.3 節のリストを元に説明する．

まず，全頂点について，リニアに抽出する場合はオブジェクトの頂点数が最大要素数を下回っている必要があり，上回った場合は偏った点群になるので注意を要する．

次にランダム抽出について，これは全頂点／分割ともに言えることだが，乱数による抽出は偏りを生じる場合もあるため，最適と思われる点群を抽出できるまで再試行することになる．この場合，分割してからランダムに抽出した方が，全頂点から直接抽出するよりも偏りを小さくできる．

最後に各データを元に分割したグループからの抽出について，本研究では $E(x_{in})$ に関係する頂点位置と法線についての分割を試行した．

この抽出法について，法線のみによる分割は頂点位置によらないため分布が偏りやすく，適切な点群を得るのは困難と判断した．また，頂点位置と法線を組み合わせた条件では，グループ数が頂点位置のみより多くなるため同分割数では最大要素数以下に収まりにくく，分割数を減らす必要があった．これにより位置による均一性を損なう結果となるため，法線を条件に加える意味は殆ど無いと判断した．

上記結果から，シミュレーションではオブジェクトの頂点数が最大要素数を下回る場合はリニアに全頂点を抽出し，上回る場合は頂点位置のグループから抽出を行った．

4.1.2 BSSRDF 拡散反射率関数テーブルの影響

本研究で提案した BSSRDF 拡散反射率関数テーブルの影響を図 4 に示す．

図の左がテーブルを使用した場合で，中央がテーブルを使用しなかった場合である．図の右にそれら画像の差分を示す．図が示すとおり，誤差はほぼ僅かである．

今回のシミュレーションではテーブルサイズを 1024 にしたが，サイズが 100 を下回るあたりから精度不足によるノイズが発生し始めることを確認している．

実際にはオブジェクトの大きさによって距離 r の取り得る範囲が異なり，それらによってノイズの影響が出るテーブルサイズは変化する．

4.1.3 描画時間比較

表 1 に，シミュレーションの結果を示す．光源数の明記されていないものは 1 光源で描画を行っている．

主に Bunny モデルを用いてシミュレーションを行い，まず比較対象として表面下散乱なしの状態で描画時間を計測した．これと 1 光源で R_d テーブルを使用したパターンと比較すると，約 3.3 秒余分に時間がかかっており，これを表面下散乱光の取得にかかった時間とした．

R_d テーブル使用の有無については約 41 秒の差があり，テーブルを使用することで約 12 倍の高速化が図れている．

2 光源，3 光源の結果については，ASL におけるサンプリング点数が最大に近くなるような位置に光源を追加した．その結果 2 光源では ASL のサンプリング点数は約 2 倍となり，表面下散乱光の取得時間も同様に約 2 倍となった．しかし，3 光源の場合は殆どが既に ASL へ追加されているサンプルに値を加算するだけとなるため，表面下散乱光の取得は 1 光源の 2.7 倍の時間となった．さらに光源を増やしてもサンプリング点数は STL 内の点数を超えないため，描画時間は収束していくこととなる．

5. お わ り に

本論文の提案する手法によって，半透明物体が従来よりも高速に描画でき，さらにハードウェア化することで実時間描画を実現することができると考える．

今後の課題としては，トーラスのような自己遮蔽物体において内部を 2 回以上通過した場合に正確な表現ができない点や，局所的な半透明性を考慮する必要のある物体への対応が

Vol.2009-CG-136 No.3
2009/8/20

図 5 (左) 鏡面反射光+拡散反射光 (右) 鏡面反射光+拡散反射光+表面下散乱光
Fig. 5 (Left) Specular + Diffuse. (Right) Specular + Diffuse + Subsurface Scattering.

あり，これらを解決することで自由形状の物体へより写実的な半透明性を適用することができるようになる．

また，実際に回路化した場合における問題の抽出と解決に加え，不均質物体の表面下散乱についても研究を進める予定である．

参 考 文 献

1) Jensen, H.W., Marschner, S.R., Levoy, M. and Hanrahan, P.: A practical model for subsurface light transport, *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, ACM, pp.511–518 (2001).
2) Jensen, H.W. and Buhler, J.: A rapid hierarchical rendering technique for translucent materials, *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, ACM, pp.576–581 (2002).
3) Dachsbacher, C. and Stamminger, M.: Translucent shadow maps, *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, pp.197–201 (2003).
4) Mertens, T., Kautz, J., Bekaert, P., Van Reeth, F. and Seidel, H.-P.: Efficient Rendering of Local Subsurface Scattering, *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, Washington, DC, USA, IEEE Computer Society, p.51 (2003).
5) Lensch, H. P.A., Goesele, M., Bekaert, P., Kautz, J., Magnor, M.A., Lang, J. and Seidel, H.-P.: Interactive Rendering of Translucent Objects, *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, Washington, DC, USA, IEEE Computer Society, p.214 (2002).
6) Carr, N.A., Hall, J.D. and Hart, J.C.: GPU algorithms for radiosity and subsurface scattering, *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, pp.51–59 (2003).
7) Schlick, C.: An Inexpensive BRDF Model for Physically-based Rendering, *Computer Graphics Forum*, Vol.13, pp.233–246 (1994).