2006－CG－122（11）
2006／2／20

# ポイントサンプルジオメトリのための表面下散乱のリアルタイムレンダリング

小田 正志† 岩崎 慶‡ 髙木 佐恵子‡ 吉本 富士市‡

†和歌山大学大学院システム工学研究科 〒640-8510 和歌山県和歌山市栄谷 930
‡和歌山大学システム工学部 〒640-8510 和歌山県和歌山市栄谷 930

E-mail: †s051015@sys.wakayama-u.ac.jp

**概要** 本稿では，大規模な点群で表現されるポイントサンプルジオメトリを用いて大理石やミルクなどの半透明物体をリアルタイムでレンダリングする手法を提案する．半透明物体のレンダリングでは，表面下散乱という物体内部での光の散乱を考慮する必要があるが，非常に計算コストが高い．そこで本稿では，物体の周囲に規則的な格子状の仮想的な輝度計算点を設置し，散乱光の輝度を各仮想点で計算し格子内部の半透明物体表面上の点の輝度は周囲の仮想点の輝度の補間により求める．この計算を GPU で行うことにより 50 万頂点からなる半透明物体をリアルタイムにレンダリングすることが可能となった．

## Real-time Rendering of Subsurface Scattering for Point-sampled Geometry

Masashi ODA† Kei IWASAKI‡ Saeko TAKAGI‡ and Fujiichi YOSHIMOTO‡

†Graduate School of Systems Engineering Wakayama University 930 Sakaedani, Wakayama, 640-8510 Japan
‡Faculty of Systems Engineering Wakayama University 930 Sakaedani, Wakayama, 640-8510 Japan

E-mail: s051015@sys.wakayama-u.ac.jp

**Abstract** We present a real-time rendering method of translucent object such as marble and milk for the point-sampled geometry. To render translucent objects, it is necessary to take into account the subsurface scattering of light. However, rendering translucent objects with the subsurface scattering takes long time. In our method, virtual points are set around the object uniformly. The intensities of the subsurface scattering of light are calculated at each virtual point. The intensities of points that represent the translucent object are interpolated by the intensities of the virtual points. Our method can render about 500,000 points with the subsurface scattering in real-time.

## 1. はじめに

近年，複雑な 3 次元形状物体を効率よく表現する手法として，ポイントサンプルジオメトリ(Point-Sampled Geometry：以下 PSG と記す)が注目されている．コンピュータグラフィックスの分野において，3 次元形状は従来ポリゴンを基本要素として表現されてきたが，PSG は点(以下サンプル点と呼ぶ)を基本要素として 3 次元形状を表現する手法である[1,4,9,12,13,14,17]．ポリゴンの代わりにサンプル点で表現する PSG の利点として，位相情報を必要としないためメモリ量がポリゴンによる表現に比べて少なくすむという点や，LOD(Level-Of-Detail)と組み合わせやすいという点が挙げられる．また，3 次元スキャナによって形状を測定した結果の点群を，ポリゴンへ再構築することなくレンダリングすることができるという利点がある．

一方近年，大理石や肌といった半透明物体を正確にレンダリングする手法種々提案されている[2,5,6,7,8,10,11,16]．これらの方法はポリゴン基本要素としたレンダリング法である．半透明物体を正確にレンダリングするためには，半透明物体内部を透過した光が物体内部で散乱し，再び物体表面から光が透過する現象(表面下散乱)を考慮する必要がある．点で表現された半透明物体の表面下散乱を考慮したレンダリング法もあるがリアルタイムにレンダリングできる方法はない．

そこで本稿では，PSG のための表面下散乱を考慮した半透明物体のリアルタイムレンダリング法を提案する．表面下散乱光を計算するためには，半透明物体表

面上のサンプル点への放射照度を計算する必要がある．半透明物体内部を散乱して透過する光は，物体表面上の各サンプル点の放射照度に，透過して光が放出する点への双方向散乱面反射率分布関数(Bidirectional Scattering Surface Reflectance Distribution Function: 以下BSSRDF と略す)を掛けたものを，物体表面の全てのサンプル点について積分することによって計算される．そのため，膨大な数の点群についての表面下散乱光の計算はコストが非常にかかる． そこで本稿では，半透明物体の各サンプル点への放射照度を， Graphics Processing Units(GPU)を利用して高速に計算する．また，半透明物体の周りに規則的な格子状の仮想点を配置し，その仮想点で表面下散乱光の積分計算を行う．各サンプル点の表面下散乱の輝度は，計算された近隣の仮想点の散乱光の輝度から補間し高速に計算する．

以下, 第 2 節では PSG および表面下散乱に関する従来法について述べる．第 3 節では表面下散乱計算について説明する．第 4 節にて提案法による表面下散乱を考慮した PSG をリアルタイムにレンダリングする手法を述べる．第 5 節において提案法で作成した結果例を示し，最後に結論および今後の課題を述べる．

## 2. 関連研究

### 2.1 Point-sampled Geometry に関する研究

PSG をレンダリングする手法は多数研究されている．点を基本要素としてレンダリングを行う方法は Levoy らによって初めて提案された[9]．Grossman らは，pull-push 法によって点群のレンダリングを行った[4]．この方法は，点群をレンダリングした際に穴ができないように，まず穴が生じない低解像度の画像を生成する．次に高解像度の画像を生成する際に，穴が生じた場合，低解像度画像を参照することで穴を埋めるものである．Pfister らは，Surfel と呼ばれる点の位置,法線,色,半径情報などを持った円盤状の基本要素で点群をレンダリングする手法を提案した[12,17]. Rusinkiewica らは，Qsplat を用いて膨大なデータ量からなる 3 次元形状を高速に表示する手法を提案した[14]．また，近年高性能化が著しい GPU を用いて点群を高速に描画する手法も提案されてきている[1,13]. これらの手法は，主に不透明な材質を取り扱っており，半透明材質の高速なレンダリング法を言及していない．

### 2.2 表面下散乱に関する研究

Jensen らは，半透明物体内部での多重散乱光を計算するための BSSRDF モデルを提案し，大理石や人の肌を正確にレンダリングした[6]．また Jensen は，階層構造を利用することで表面下散乱光を高速に計算する手法を提案した[7]．Lensch らは，視点および照明条件を変化してもインタラクティブに半透明物体をレンダリングする手法を提案した[8]．Hao らは，局所的な表面下散乱光のみを考慮することでリアルタイムに半透明物体をレンダリングする手法を提案した[9]．Carr らは，物体のテクスチャアトラスを生成し，GPU を利用してテクスチャ上で表面下散乱の計算を行う手法を提案した[2]．Mertens らは，透明物体の幾何形状が変化しても高速に表面下散乱を計算する手法を提案した[10][11]．Dachsbacher らは，シャドウマップのように光源からレンダリングされた輝度テクスチャを GPU 上でサンプリングする手法を提案した[3]. Wang らは，全周波数を前計算することにより表面下散乱をインタラクティブな速度でレンダリングする手法を提案した[16].

しかしながら，これらの方法はポリゴンで表現された 3 次元形状に対しての計算法であり，また頂点数が少ない物体に対しての手法である．これらの方法を直接 PSG に適用することは難しい. PSG のための表面下散乱の研究としては文献[15]があげられるが，この方法はモンテカルロ法を使用して表面下散乱を計算しており，計算コストが非常に高い．また我々の文献[18]での提案手法では計算した表面下散乱光を CPU に一度読み直す必要があり，リアルタイムレンダリングを達成することができなかった. そこで, 本稿では, PSG のための表面下散乱のリアルタイムレンダリング手法を提案する．

## 3. 表面下散乱の計算の概要

提案手法では，表面下散乱の計算に関して Jensen らが提案した拡散近似による BSSRDF モデル[6]を利用する．

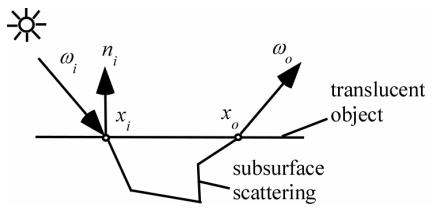



図 1: BSSRDF モデル[6]

本節ではこの BSSRDF モデルについて簡単に述べる．半透明物体上のサンプル点 $x_o$ から $\omega_o$ 方向に放出する表面下散乱の光の輝度 $L_o(x_o,\omega_o)$ は以下の式で計算される(図 1 参照).

$$L_o(x_o,\omega_o)=\int_A\int_\Omega L_i(x_i,\omega_i)S(x_i,\omega_i,x_o,\omega_o)(n_i\cdot\omega_i)d\omega_i dA(x_i) \quad (1)$$

ここで $A$ は物体表面の面積，$\Omega$ はサンプル点 $x_i$ における半球上に分布した方向，$L_i(x_i,\omega_i)$ はサンプル点 $x_i$ における $\omega_i$ 方向からの入射輝度，$S(x_i,\omega_i,x_o,\omega_o)$ は BSSRDF，$n_i$ はサンプル点 $x_i$ における法線，$dA(x_i)$ はサンプル点 $x_i$ における微小面積とする．Jensen らは，BSSRDF を以下の拡散成分 $S_d$ で近似して計算した．

$$S_d(x_i,\omega_i,x_o,\omega_o)=\frac{1}{\pi}F_t(\eta,\omega_i)R_d(x_i,x_o)F_t(\eta,\omega_o) \quad (2)$$

ここで，$F_t$ はフレネル項，$\eta$ は半透明材質の相対屈折率ある．$R_d$ は拡散反射率で次の式で計算される．

$$R_d(x_i,x_o)=\frac{\alpha'}{4\pi}\left[z_r(1+\sigma_{tr}d_r)\frac{\exp(-\sigma_{tr}d_r)}{d_r^{\,3}}\right]+\frac{\alpha'}{4\pi}\left[z_v(1+\sigma_{tr}d_v)\frac{\exp(-\sigma_{tr}d_v)}{d_v^{\,3}}\right] \quad (3)$$

ここで，式(3)で用いられている変数は以下の式(4)～(13)で計算される．

$$z_r=1/\sigma'_t \quad (4)$$

$$z_v=z_r+4AD \quad (5)$$

$$d_r=\left\|x_i-z_r n_i-x_o\right\| \quad (6)$$

$$d_v=\left\|x_i+z_v n_i-x_o\right\| \quad (7)$$

$$A=\frac{1+F_{dr}}{1-F_{dr}} \quad (8)$$

$$F_{dr}=-\frac{1.440}{\eta^2}+\frac{0.710}{\eta}+0.668+0.0636\eta \quad (9)$$

$$D=1/3\sigma'_t \quad (10)$$

$$\sigma_{tr}=\sqrt{3\sigma_a\sigma'_t} \quad (11)$$

$$\sigma'_t=\sigma_a+\sigma'_s \quad (12)$$

$$\alpha'=\sigma'_s/\sigma'_t \quad (13)$$

ここで $\sigma_a$ は半透明材質の光の減衰率，$\sigma'_s$ は縮小散乱係数とし，材質特有のパラメータとして与えられる[6].

## 4. 提案法

### 4.1. 提案法の概要

提案法では，入射光は平行光源とする．点群は非常に細かくサンプリングされたものとし，各サンプル点はその位置情報 $x_i$，法線ベクトル $n_i$ が既知とする．提案法では，全てのサンプル点について同じ半径 $r_i$ および同じ面積 $A_i$ が割り当てられているものとする．

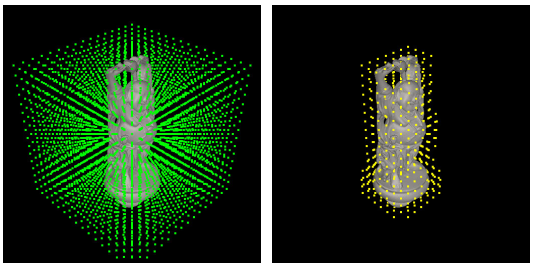
式(1)に式(2)を代入し，以下の式で表面下散乱光を計算する．

$$L_o(x_o,\omega_o)=\frac{F_t(\eta,\omega_o)}{\pi}\int_A E(x_i)R_d(x_i,x_o)dA(x_i) \quad (14)$$

$$E(x_i)=\int_\Omega L_i(x_i,\omega_i)(n_i\cdot\omega_i)F_t(\eta,\omega_i)d\omega_i \quad (15)$$

式(15)は物体表面上の各サンプル点の入射輝度にフレネル項を掛けて半球上で積分したものである．サンプル点 $x_o$ から視点への表面下散乱光の放射輝度 $L(x_o,\omega_o)$ は，入射光によって照らされている物体表面上のサンプル点 $x_i$ の放射照度 $E(x_i)$ を，物体表面上で積分することによって計算される．しかし，膨大な量の点群からなる半透明物体の場合，全ての点に対して光に照らされているかを判定し，放射照度を計算するのは非常に計算コストが高い．そこで，本稿では，文献[3]と同じようにシャドウマップ法のように光源を視点として物体をレンダリングし，その結果をテクスチャとして保存する．そのテクスチャの各ピクセルに対応する物体表面上のサンプル点において $E(x_i)$ を計算する．

また全ての可視点 $x_o$ に対して式(14)から表面下散乱を計算することは計算コストが非常に高い．しかし，近隣のサンプル点同士では表面下散乱光成分にあまり差がないと考えられる．そこで可視点全てについて表面下散乱の輝度を計算する代わりに，物体を囲むような立方体を考えその中に規則的に格子状の仮想点を配置し（図 2(a)），この各仮想点において表面下散乱を計算する．物体を構成する各サンプル点の表面下散乱光の輝度は，近隣の８つの仮想点の表面下散乱光の輝度を線形補間することによって求める．これによって約 50 万程度の点群で構成される半透明物体についてリアルタイムでレンダリングすることができる．なお点群のレンダリング方法は文献[1]の手法を利用した．



(a)物体を囲む立方体内に配置された仮想点　　(b)近隣に物体が存在しない仮想点を除去

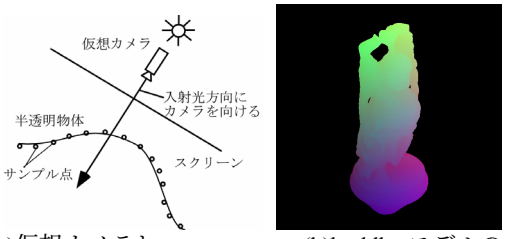
図 2:仮想点の配置

### 4.2. 放射照度計算

数十万もの点群から構成される PSG において全ての点で光が直接当たるか判定していては，計算コストが大きい．また光に照らされている全ての点において式（14）の積分計算を行いリアルタイムでレンダリングすることは難しい．そこで本稿では，次の方法で半透明物体の放射照度を効率的に計算する．

図 3(a)に示すように入射光の向きと同じ向きに仮想カメラを設置する．本稿では光源を平行光源に限定しているため，スクリーンに半透明物体全体が投影されるように光源と物体の距離を一定に設定し，仮想的なカメラから物体をレンダリングする．これによって光が当たっている点を判定することができる．仮想カメラから物体をレンダリングした画像をテクスチャとし

て保存する．このテクスチャを illumination map と呼ぶ．テクスチャの各ピクセルに投影されたサンプル点について放射照度を計算する．この点を放射照度サンプル点と呼ぶ．このテクスチャの各ピクセルの RGB 値に放射照度サンプル点の $xyz$ 座標を $0 \le x, y, z \le 1$ に正規化して保存し，α 値に放射照度サンプル点での照度の値を保存する（図 3（b））．また放射照度サンプル点がないピクセルの α 値は 0 で初期化しておく．テクスチャの生成は GPU の頂点プログラムを利用することで高速に処理される．




(a)仮想カメラとスクリーン　(b)buddha モデルの illumination map

図 3：illumination map の求め方と例

### 4.3. 仮想点における表面下散乱の計算

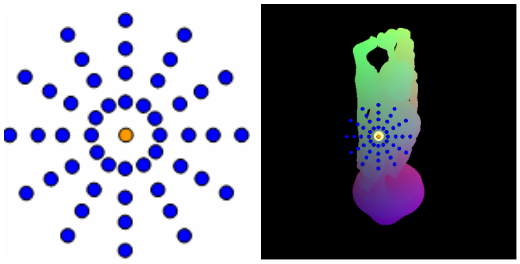
サンプル点 $x_o$ から視点に放射する表面下散乱光の輝度 $L_o(x_o, \omega_o)$ は，放射照度サンプル点の照度を用いて以下のように計算される．

$$L_o(x_o, \omega_o) = \frac{F_t(\eta, \omega_o)}{\pi} \sum_N E(x_i) R_d(x_i, x_o) \Delta A(x_i) \qquad (16)$$

ここで，$N$ は放射照度サンプル点の数とし，$\Delta A(x_i)$ は放射照度サンプル点に割り当てられた面積とする．視点から見えるサンプル点全てに対して式(16)を計算すると，膨大な数の点群からなる物体では計算コストが高い処理となる．そこで本稿では，半透明物体の周りの規則的な格子状の仮想点を配置し，仮想点で表面下散乱光を計算し，格子内部のサンプル点における表面下散乱光の輝度は仮想点における輝度から補間して求める．各仮想点はオブジェクトのまわりに縦横奥行き方向に 2 の累乗個配置し（図 2(a)），近隣にオブジェクトが存在するものだけ残す（図 2(b)）．この各仮想点 $x_{virtual}$ において，以下の輝度 $B(x_{virtual})$ を計算する．

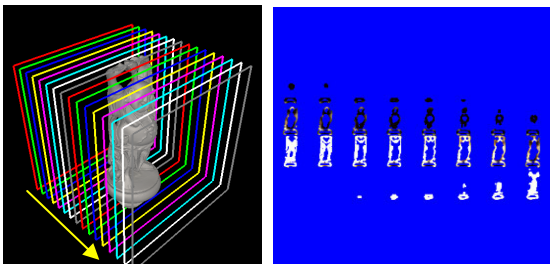
$$B(x_{virtual}) = \frac{1}{\pi} \sum_N E(x_i) R_d(x_i, x_{virtual}) \Delta A(x_i) \qquad (17)$$

表面下散乱光を計算する際は，仮想点を光源位置のカメラからレンダリングし，仮想点の位置情報と illumination map の RGB 値（放射照度サンプル点の $xyz$ 値）とを比較する．その際全ての放射照度サンプル点からの寄与を計算するのは計算コストが高いので，式(17)をガウスの正規分布を考慮した 49 点からなるサンプリングパターン（図 4(a)）を用いて，GPU で放射照度サンプル点から仮想点への寄与を高速に計算する．このときサンプリングパターンは illumination map の大きさ 512x512pixel に対して 40x40pixel とした（図 4(b)）．仮想点は図 5(a)のように $z$ 方向に対して分割数に応じてスライス式にサンプル面にグループ分けされる．サンプル面上の仮想点の輝度を，サンプル面ごとにタイル上に並べて 1 枚のテクスチャとして保存する（図 5(b)）．このテクスチャを仮想点輝度テクスチャと呼ぶ．図 5(b)は buddha モデルに対し，後ろ方向から光を当てた場合を表している．



(a)サンプリングパターン　(b)仮想点での表面下散乱の輝度計算

図 4:GPU でのサンプリングパターンを用いた仮想点での表面下散乱光の輝度の計算



(a)サンプル面　(b)仮想点輝度テクスチャ

図 5:仮想点のグループ分けとテクスチャへの保存

### 4.4. サンプル点の表面下散乱の計算

各サンプル点 $x_i$ の表面下散乱光の輝度は，仮想点の輝度 $B(x_{virtual})$ を保存した仮想点輝度テクスチャを利用し，近隣の仮想点の輝度から線形補間で求める．サンプル点の $z$ 値から，隣接する 2 枚のサンプル面を判定する．サンプル点の $xy$ 値から，サンプル点を隣接するサンプル面に投影した点のテクスチャ座標を計算する（図 6）．これにより，サンプル面に投影した点の輝度を仮想点輝度テクスチャから得ることができる．サンプル点の輝度は，投影した 2 点の輝度をサンプル点の $z$ 値と 2 枚のサンプル面の $z$ 値を用いて補間する．最終的に，各サンプル点でのフレネル項を掛けて輝度を計算する．これらの計算を全て GPU で行うことで高速に計算する．

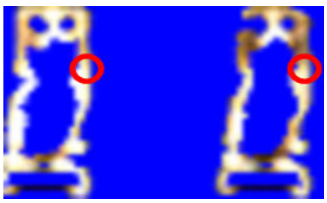


図 6:隣接したサンプル面に投影したサンプル点（○で囲まれた部分）

図 7:材質が大理石の buddha モデル

図 8:材質が大理石の teapot モデル

図 9:材質がミルクの dragon モデル

提案法では，高速化のためにサンプル点での表面下散乱の輝度計算を仮想点での輝度の補間で求めている．そのため全てのサンプル点で表面下散乱を計算した場合とでは誤差が生じる．図 10 は提案法による画像と全サンプル点で輝度計算した生成画像である．各ピクセルの色の誤差は最大で 10.72%，平均で 1.57%であり補間による計算を行っても画質にほとんど差が見られない．図 10(b)の描画時間は約 517.2 秒であり，提案法によって約 20,000 倍高速に描画することが可能となった．

(a)提案法　　(b)全サンプル点で輝度計算

図 10:提案法と全サンプル点で輝度計算した画像

表 1:提案手法の計算時間

| 物体 | 頂点数 | 仮想点数 | FPS |
|---|---|---|---|
| buddda(図 7) | 505,576 | 13,738 | 40.25 |
| teapot (図 8) | 454,990 | 13,610 | 43.36 |
| dragon(図 9) | 640,826 | 18,737 | 30.97 |

## 5. 結果

本稿で提案した手法を用いて表面下散乱を計算した適用例を図 7 から図 10 に示す. 図 7 は大理石の buddha モデル（サンプル点数:505,576）である．図 8 は大理石の teapot モデル（サンプル点数:454,990）に大理石調のテクスチャを貼り付け，セルフシャドウを考慮したものである. 図 9 は材質をミルクとした dragon モデル（サンプル点数:640,826）である．計算時間を表 1 に示す．このように，提案法によって 50 万点以上の点群からなる膨大な PSG を，表面下散乱を考慮してリアルタイムでレンダリングすることができた．実行環境は CPU が Pentium4 2.0GHz,GPU が GeForce6800 Ultra の PC である．プログラムは，C++，OpenGL，Cg で実装した．画像サイズは全て 512x512 である．

## 6. まとめと今後の課題

本稿では，PSG のための表面下散乱のリアルタイムレンダリング法を提案した．まず物体の周りに格子状の仮想点を配置し，その仮想点で表面下散乱を計算する．物体を構成するサンプル点での表面下散乱の輝度は，近隣の仮想点から補間して計算した．提案法により，約 50 万の点群からなる半透明の物体を，表面下散乱を考慮してリアルタイムでレンダリングすることができた．

今後の課題として以下が挙げられる．現在単一の平行光源しか考慮にいれていない．そこで線光源や面光源または複数の光源への適応が考えられる．また提案手法では前計算を必要としない．そのため物体の形状が動的に変化しても適用可能だと考えられるため，変形物体への適用も今後の課題である．

## 参考文献


[1] M. Botsch and L. Kobbelt, “High-Quality Point-Based Rendering on Modern GPUs”, Proc. of Pacific Graphics 2003, pp. 335-343, October 2003.

[2] N. A. Carr, J. D. Hall, and J. C. Hart, “GPU Algorithms for Radiosity and Subsurface Scattering”, Proc. of Graphics Hardware 2003, pp. 51-59, July 2003.

[3] C. Dachsbacher, and M. Stamminger, “Tlanslucent Shadow Maps”, Proc. of Eurographics Symposium on Rendering 2003, pp. 197-201, June 2003.

[4] J. P. Grossman, “Point Sample Rendering”, Master’s thesis, Department of Electical Engineering and Computer Science, MIT, August 1998.

[5] X. Hao, T. Baby, and A. Barshney, “Interactive Subsurface Scattering for Translucent Meshes”, Proc of. Symposium on Interactive 3D Graphics 2003, pp. 75-82, April 2003.

[6] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanraham, “A Practical Model for Subsurface Light Transport”, Proc. of SIGGRAPH 2001, pp. 511-518, August 2001.

[7] H. W. Jensen and J. Buhler, “A Rapid Hierarchical Rendering Technique for Translucent Materials”, Proc. of SIGGRAPH 2002, pp. 576-581, July 2002.

[8] H. P. A. Lensch, M. Goesele, P. Bekaert, J. Kautz, M. A. Magnor, J. Lang, and H. P. Seidel, “Interactive Rendering of Translucent Objects”, Proc. of Pacific Graphics 2002, pp. 214-222, October 2002.

[9] M. Levoy and T. Whitted, “The Use of Points as Display Primitives”, Technical Report, TR 85-022, The University of North Carolina at Chapel Hill, Dept. of Computer Science, 1985.

[10] T. Mertens, J. Kautz, P. Bekaert, F. V. Reeth, and H. P. Seudel, “Efficient Rendering of Local Subsurface Scattering”, Proc. of Pacific Graphics 2003, pp. 51-58, October 2003.

[11] T. Mertens, J. Kautz, P. Bekaert, F. V. Reeth, and H. P. Seudel, “Interactive Rendering of Translucent Deformable Objects”, Proc. of Eurographics Symposium on Rendering, pp. 130-140, June 2003.

[12] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, “Surfel: Surface Elements as Rendering Primitives, “Proc. of SIGGRAPH 2000, pp. 335-342, July 2000.

[13] L. Ren, H. Pfister, and M. Zwicker, “Object Space EWA Splatting: A Hardware Accelerated Approach to High Quality Point Rendering”, Computer Graphics Forum, Vol.21, No.3, pp. 461-470, September, 2002.

[14] S. Rusinkiewicz and M. Levoy, “QSplat : A Multiresolution Point Rendering System for Large Meshes”, Proc. of SIGGRAPH 2000, pp. 343-352, July 2000.

[15] G. Schaufler and H. W. Jensen, “Ray Tracing Point Sampled Geometry, “Proc. of Eurographics Workshop on Rendering, pp. 319-328, June, 2000.

[16] R. Wang, J. Tran, D. Luebke, “All-Frequency Intractive Relighting of Translucet Objects with Single and Multiple Scattering”, Proc. of SIGGRAPH2005, pp. 1202-1207, July 2005.

[17] M. Zwicker, H. Pfister, J. van. Baar, and M. Gross, “Surface Splatting”, Proc. of SIGGRAPH 2001, pp. 371-378, August 2001.

[18] 小田正志, 岩崎慶, 髙木佐恵子, 吉本富士市, “ポイントサンプルジオメトリのための表面下散乱の高速計算法”, 画像電子学会第 216 回 pp. 41-46, March 2005


## 謝辞


本稿で使用した”buddha”モデル，”dragon”モデルは G. Turk, B. Mullins の両氏のご好意で利用させていただきました．