イメージベースドポイントレンダリングによるレンジスキャンデータの描画

川田弘明*　金井崇*
* 慶應義塾大学 環境情報学部
E-mail: {t02282hk/kanai}@sfc.keio.ac.jp

本稿では、複数のレンジスキャンデータを描画するためのイメージベースドポイントレンダリング (IBPR) の手法を提案する。 pull-push と呼ばれる手法を拡張することで、複数のレンジデータが計測時の誤差を含む場合においても、滑らかな表面を描画することができる。そして、描画におけるブラーを軽減し、高品質な描画結果を得る方法をも提案する。本手法は、レンジスキャンデータのスキャンの際のラフ形状チェック等に有効である。

Image-Based Point Rendering for Multiple Range Data

HIROAKI KAWATA*　TAKASHI KANAI*
*Keio University, Faculty of Environmental Information
E-mail: {t02282hk/kanai}@sfc.keio.ac.jp

In this paper, we describe image-based point rendering (IBPR) for multiple range data from 3D scanners. Our approach is a natural extension of the method called pull-push so as to render scanned points with some measurement errors. Several extensions for rendering range data is proposed. One is a seamless rendering of a whole object even for points that have measurement errors. The other is a high quality rendering to reduce blurring. Our method is suitable for roughly checking the shapes only by using range scan.

1　はじめに

近年、 3 次元測定機は改良され詳細なモデルを得ることができるが、データ量は多くなるという傾向にある。また、 3 次元データによって得られたデータからメッシュを得るために、点群から 3 次元形状を再構築する技術が研究されている。

一般的に点を描画するという事は、ポリゴンを描画することに比べて高速であることから、 LOD (Level-Of-Detail) と組み合わせたポイントレンダリングの手法が提案されている。例として、地形 [1] を描画する場合、エコシステム [2]、複雑なオブジェクト [3] などに対しての応用などが議論されている。

ポイントレンダリングにおいての大きな利点として、ポリゴンを描画する場合に必要な面の接続情報が必要ないという事が挙げられる。しかし、接続情報を必要としないという事から、点と点との間の隙間をどのようにして埋めるのかという事が同時に大きな問題となる。点と点との間の隙間を埋める方法として、主に *splat* による方法や pull-push による方法がある。

Rusinkiewicz と Levoy は QSplat[4] と呼ばれる手法によって 3 次元測定機によって得られた大量の点群データを描画する方法を提案している。 QSplat は、多重解像度表現が可能なデータ構造によって視点から見える点群を効率よく検索し、必要な部分のみを描画することによって、高速な描画を可能としている。また、 QSplat で必要とするデータ構造は前処理によって処理しておく必要がある。

Zwicker ら [5, 6] はポイントレンダリングにおいて、高品質な *surface element* (*surfel*) を用いて点群を描画する手法を提案している。 Surfel[5] とは、点の位置、法線、色情報、半径などによって定義される描画の際に必要な要素を表している。 Surfel を描画するにあたり Splat を使用し、 4 角形、円などの形状を点の代わりに描画することによって物体を描画する。また、 Zwicker らは Surfel によるアンチエリアシングの手法も提案している。

Grossman と Dally[7] は、 *pull-push* と呼ばれる手法によって点群の描画を行っている。この手法は、点と点の間の穴を埋める手法として、穴が存在する画像から低解像度の穴が埋まった状態の画像を生成し、元画像の穴があいている部分は低解像度の画像を参照して穴を埋めるという手法を利用したものである。

本手法は、三次元測定機によって得られた点群を前

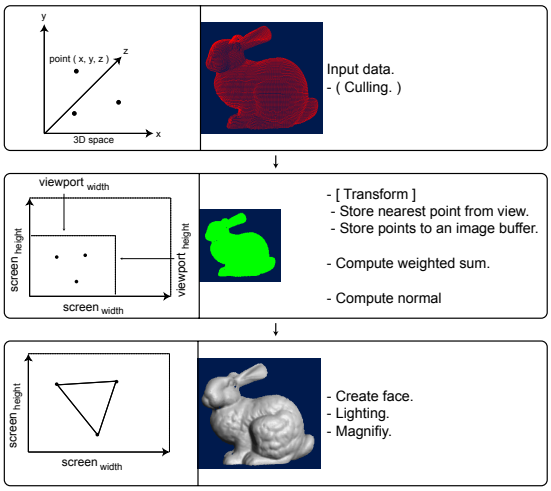


図 1: 描画の手順

処理を必要とせずに直接描画する手法を提案する。本手法はイメージベースドレンダリングによる方法を基本としていることからイメージベースドポイントレンダリング (Image-Based Point Rendering : IBPR) と呼ぶことにする。本手法では、点を描画することにおいて発生する点と点との穴を pull-push による性質を基本として穴を埋めるという処理を行っている。また、本手法は 3 次元測定機において得られた複数のレンジデータの計測の誤差によって発生するオーバーラップを含むデータにおいても、前処理を必要とせず高品質に描画することができる。 また、本稿では、IBPR における物体を半透明で描画する手法も同時に提案する。

2 IBPR の描画ステップ

この章では IBPR に必要なデータと描画方法について説明する。また、 IBPR においての半透明で物体を描画する手法についても提案する。

2.1 アルゴリズム

IBPR は、レンジスキャンデータを対象とした描画手法であり、最低限必要なデータとして、点群を表す (x, y , z) の 3 次元座標データ、そして描画するデータの解像度のデータが必要となる。また、描画する際に色の情報が必要であれば、 3 次元の頂点座標にそれぞれ対応した色情報 (r, g, b, a) の情報も必要となる。シェーディングをする際に必要な法線情報は用意する必要はなく、毎フレーム計算を行う。 図 1 は、IBPR の描画において毎フレーム行う処理をまとめたものである。これらの処理は、

1. 描画するデータからスクリーンの解像度の大きさを計算する

2. 点群のデータを用意したバッファに書き込む

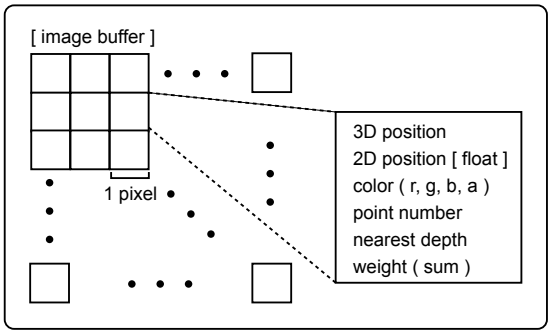


図 2: イメージバッファ

3. バッファから法線を計算し、シェーディングの処理を行う

4. 描画するスクリーンの大きさに拡大しバッファを参照して面を描画する

のような手順となる。また、バッファには図 2 のような情報を記録する。

出力画像の品質を更に向上させる方法として、後のセクションで説明する 2 パスによる描画方法がある。バッファに記録する際の解像度 $(Viewport_{width}, Viewport_{height})$ は、以下の式によって計算を行う。

$$\begin{aligned} Viewport_{width} &= \frac{Screen_{width}}{S}, \\ Viewport_{height} &= \frac{Screen_{height}}{S}, \end{aligned} \tag{1}$$

$S(S \geq 1)$ は、視野角、視点の位置、物体の位置、点群の解像度によって決定される。また、$(Screen_{width}, Screen_{height})$ は結果の画像の幅と高さを表している。

2.2 バッファへの記録

描画する物体の点群に対して、座標変換、投影変換を行い、Sによってスケーリングを行う。また、頂点はラスタライズされた座標の位置のイメージバッファに保存される。もし、イメージバッファにおいて同じピクセルに他の点の値が入るのであれば、それぞれの点の情報の重み付平均和を計算する。また、重み (w) は以下の式によって計算を行う。

$$w = \frac{|z - z_{near}|}{\gamma}, \tag{2}$$

式 (2) において z_{near} は視点から最も近い点の距離を表している。また、z は点のデプス値を表している。γ は、物体の表面の面とみなすしきい値を表している。この、しきい値は複数のレンジデータを描画する際に重要な値であり、また、γ の値については図 3 に示している通りである。もし、z_{near} から $z_{near} + \gamma$

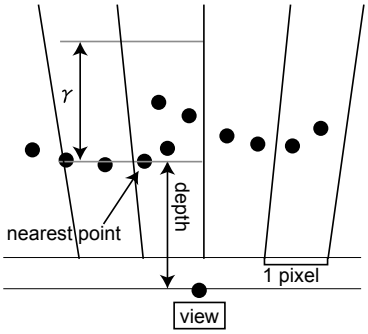


図 3: γ の値.

より遠くに点がある場合には、バッファにはその点の値は記録しない事にする。

また、γ は複数のレンジデータを描画する際において、γ を設定することにより、データが計測の際のエラーを含んでいる場合に誤差を含めて表面を描画することができるため便利である。γ の値は、実験の結果レンジデータの解像度の 1 から 3 倍程度の値が有効である。

2.3 法線の生成

本項では、シェーディングの計算に必要な法線の生成方法について述べる。点群に関しての法線の生成方法は Hoppe らの研究 [8] で紹介されている。この方法は、それぞれの点に関して近傍の点群を集め慣性楕円体の主軸を計算する。この方向は、近似された法線ベクトルと平行な軸として考える。しかし、この方法は前処理で計算しておく必要があり他の方向からスキャンしたレンジデータも合成した状態で処理を行う必要がある。本章で説明する法線の生成方法は物体を描画する過程で計算を行い、更に計算方法は簡単であるという利点を持つ。

本手法では、近傍のピクセルを参照して法線を生成する。近傍のピクセルを参照し値を持っているのであれば面を生成し法線を計算する。近傍のピクセルを参照して面を生成するパターンは図 4 に示している通りである。図 4 では、近傍の 8 つのピクセルのうち一つの面が生成できるかを判定するために 2 つずつピクセルを参照して、面が生成できるようであれば面を生成し法線を計算する。複数の面が生成できるようであれば、全てパターンを計算して平均値を計算して値を割り当てる。

しかし、この方法は、近傍のピクセルのデプス値が離れている場合には図 5(a) のようにぼやけた画像が生成されてしまうという問題を持つ。このような問題を避けるために、デプス値の距離のしきい値を設定し、デプス値が離れすぎている場合には法線の計算を行わないようにする。また、このしきい値は物体の解像度の 5 倍程度を本研究では割り当てている。しきい値を使用した場合の描画結果を図 5に示す。図 6は、法線を生成するためのピクセルの参照方法について示している。しきい値を使用している図の右側では、

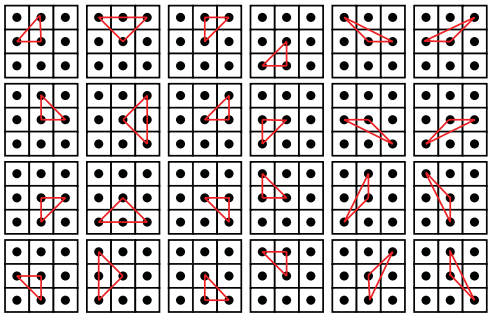
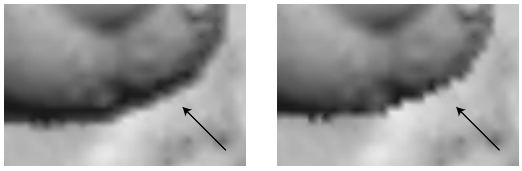


図 4: 面のパターン (法線計算用)



(a) (b)

図 5: しきい値を設定した場合としていない場合の結果、 (a) しきい値を設定していない場合 (b) しきい値を設定した場合。

デプス値が離れすぎているピクセルを参照する場合には、法線の計算をしていない事を示している。

2.4 面の生成

本稿では、面を近接するピクセルを参照して生成する。この過程を図 7 に示す。まず、近傍のピクセルを参照する。図においては、参照するピクセルは、右、下、斜め右下のピクセルを参照している。もし、近接するピクセルを参照して 3 つのピクセルが値を持っているのであれば 3 角面を生成し、 4 つであれば 4 角面を生成する。

面が生成できたのであれば、 S を使用してスクリーンの大きさに拡大して表示を行う。また、面を生成するそれぞれの頂点に対して、色を割り当てる必要がある。また、同時にシェーディングの計算も行う。

しかし、面を生成し 2 次元の座標を決定する際に、ラスタライズされたバッファの位置 (整数値) を利用してを使用してしまうと図 8 の上に示されるように面の形状がうまく復元されない。このようなことから、頂点に割り当てる 2 次元の座標の値は浮動小数点の値を使用する事にする。この値は 3 次元から 2 次元に投影された状態の浮動小数点の値をそのまま使用することとなる。浮動小数点の値を使用すると図 8 の下に示すように、整数値を使用した場合に比べて元の形状を復元することができる。

2.5 鮮鋭化の処理

本手法の問題点として結果の画像がぼやけてしまうという事が挙げられる。図 10 は、この問題を示したものである。この問題は、シルエットエッジの部分

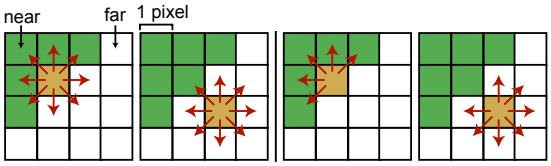


図 6: 法線を生成する面

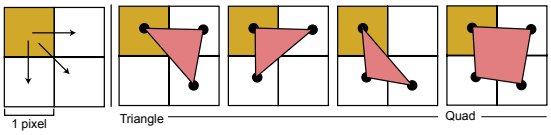


図 7: 面の生成

(図 10(a) の左) や視点に対して法線が垂直となるような部分 (図 10(b) の左) に発生する。

この問題を解消するために 2 パスの描画方法が考えられる。まず、 1 パス目では、 Sの大きさを 1 パスのみで描画する場合に比べて大きめに設定して描画を行う。エッジに関しては図 9 に示されるように点の密度が高いため S の値を大きく設定している。 S は 1 パスのみの描画の場合に比べて 0.83 から 0.66 倍程度の値が望ましい。

2 パス目では、通常の S の解像度で描画を行う。 1 パス目の描画によって物体が描画された後に処理を行い、色は上書きする。シルエットエッジに関しては、 2 パス目の結果を上書きすることによりシャープな結果を得ることができる。しかし、視点に対して法線が垂直となるような部分に関しては、 2 パス目の結果を反映しないようにする必要がある。 2 パス目の解像度で描画されてしまうと、この部分の結果は 2 パス目の解像度となってしまうためである。このような事を避けるために、 2 パス目の描画において次に示すような部分では面を描かないようにする。

1. ピクセルにおいて点の密度が高い場合 (しきい値を設定しておく: 8 点程度が望ましい)

2. 視点に対して法線が垂直となるような部分

図 10 (a) の右ではシルエットエッジの例であり、 2 パスで描画した結果の方が滑らかなシルエットエッジを描画することができていることが分かる。図 10 (b) の右では、視点に対して法線が垂直となるような部分のエッジでの結果を示したものであり、この場合も 2 パスで描画した結果が 1 パスでの描画に比べてより滑らかなエッジを描画できていることが分かる。

2.6 半透明の処理

本章では IBPR による描画で使用する半透明で物体を描画する手法について説明する。

半透明の描画の方法は、 Everitt によって提案されているものがあり、これは複数のパスによって裏にある面を並び替えて半透明の物体を描画する [9] という手法である。

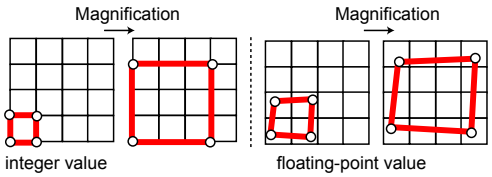


図 8: 浮動小数点型の値を使用して拡大した場合

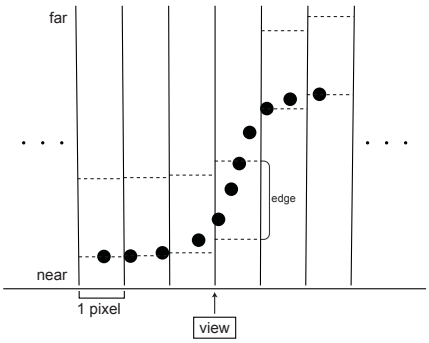


図 9: ピクセルに記録される点の数

IBPR 用の半透明の描画方法は、複数のレイヤーを生成するためのイメージバッファを使用する。図 11 では、物体を描画する際のレイヤーの生成方法を示したものである。それぞれのピクセルは複数のレイヤー持っており、この各レイヤーには、 2 次元の頂点座標、深さ、色、 3 次元座標を持っている。また、複数の頂点の値が入る場合には、各値の平均値を計算する。また、半透明の描画の手順は以下に示す通りとなる。また、レイヤーの深さの範囲 ($threshold_{layer}$) は、本研究では物体の解像度の 10 倍に設定し、レイヤーは 5 つ生成して使用している。

1. 視点に一番近い点と判断される点以外に対して、全てのレイヤーのデプスの値と点のデプスの値を比較し、どのレイヤーに対してもデプス値が離れている場合には、新たなレイヤーを作成し、そこに値を書き込む。既に生成されたレイヤーの範囲に点がある場合は、色、 3 次元座標に対して平均値をとることで値をアップデートする。

2. 全ての点に対して処理が行った後に、各ピクセルのレイヤーのデプス値が視点に近い順にレイヤーをソートする。また、各レイヤーに対して色と 3 次元座標の平均値を計算する。

3. 各レイヤーに対して、法線を生成してライティングの計算を行う。

4. 各レイヤーの色をそれぞれ合成する。色の合成は式 (3) の通りとなる。 c_{src} は元のピクセルに記録されている色、 c_{dst} はこれから合成する色、 c は合成後の色、 $\alpha(0 \leq \alpha \leq 1)$ は透明度を示す。また、この計算は RGB の各要素に対して行う。

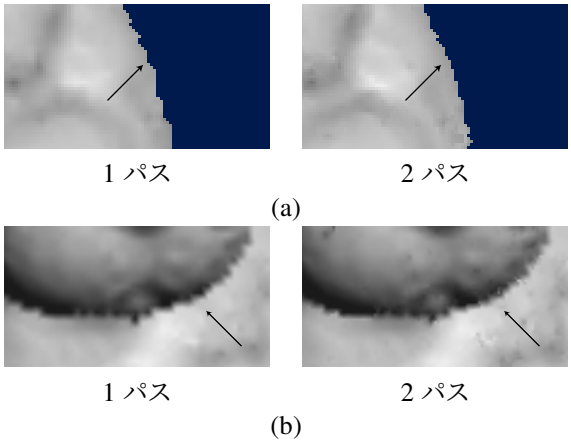


図 10: 異なるパス数での描画結果 (a) シルエットエッジ、 (b) 視点の方向と法線が垂直に近い部分

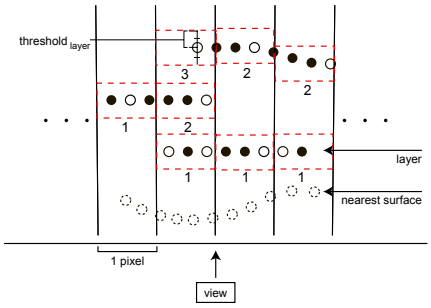


図 11: 半透明用のレイヤー

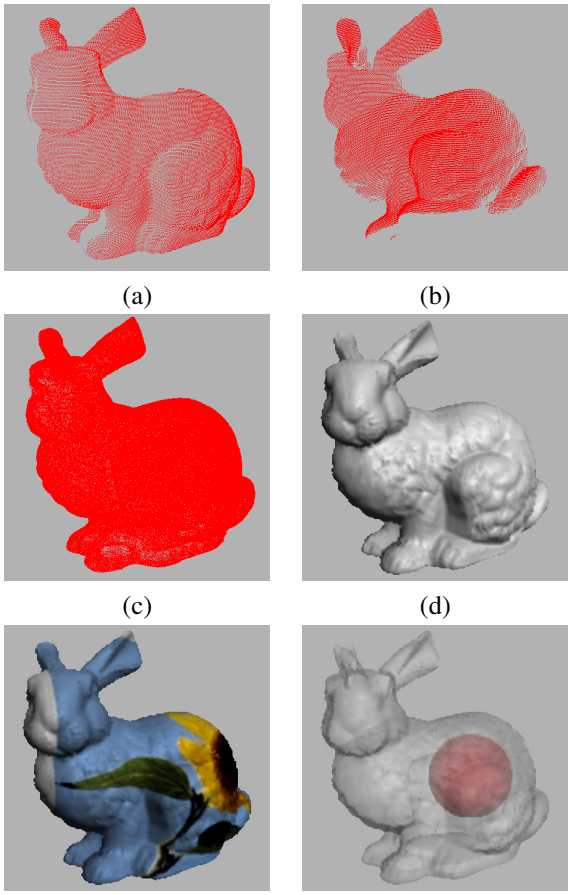


図 12: “Bunny” の描画結果

$$c = c_{dst}(1-\alpha) + c_{src}\alpha \tag{3}$$

2.7 カリングの処理

IBPR の一つの特徴として、複数のレンジデータを描画する際に全てのレンジデータを合成してから描画する必要はない。このような特徴から、必要なレンジデータのみを描画することによって、描画速度の向上と、より手軽なレンジデータのチェックを行うことができる。

ここでは、レンジデータ単位でのカリングの方法を述べる。一つのレンジデータは一つの方向からスキャンされたものである。このような条件から、スキャン時のレンジデータの情報によってカリングを行う。カリングを行うには、対象とするレンジデータのスキャンされた時の方向と、視点の向きを使用して可視判定を行う。可視判定は、バックフェースカリングと同じ方法を用いる事によって判定を行うことができる。

3 結果

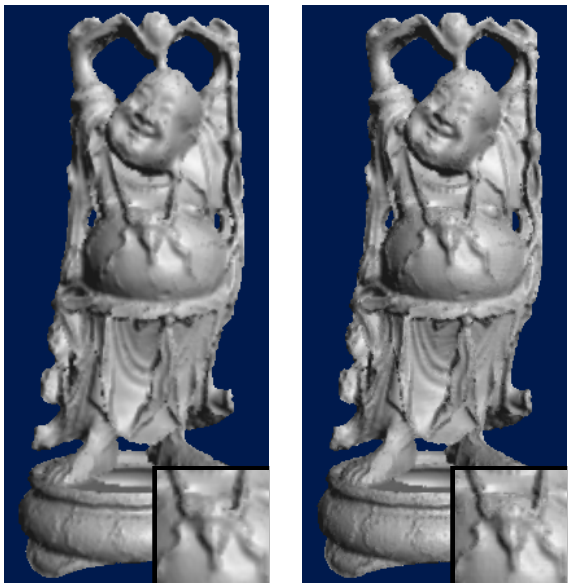
本研究において、二つのレンジデータを使用して本手法の評価を行った。計算環境は Pentium4 2.53GHz の CPU を使用した。また、評価用に使用したソフトウェアは、面の描画の部分には OpenGL を使用したが、座標変換などのそのほかの処理は全て CPU で計算を行っている。

一つの評価用のデータとして ” Bunny ” のレンジデータを使用した。図 12 は ” Bunny ” を IBPR で描画した結果である。図 12 の (a) と (b) は、それぞれスキャンした方向の違う一つのレンジデータを描画したものである。また、 (c) は 10 方向からスキャンしたデータ (362,272) を合成したものを描画している。図 12 の (d) は 1 パスで図 12 (c) のデータを IBPR で描画した結果である。

図 12 の (e) は図 12 (c) のデータに色を加えて描画したものであり、図 12 の (f) は図 12 (c) のデータに球を ” Bunny ” の物体の中に合成して半透明描画したものである。

また、二つ目の描画するデータとして Happy Budda(1,274,573) を使用した。このデータは 18 の方向からスキャンしたデータを合成したものである。図 13は 1 パス、 2 パスでの描画結果である。

描画時間は表 1 の通りである。表の、 model は使用したモデルの種類、 S は、バッファに記録する際の解像度 ($Viewport_{width}, Viewport_{height}$) を決定する際の S の値、 Alpha は半透明の処理の有り (yes)、無し (no)、 Point は、点の数、 Pass は描画全体のパス



(a) (b)

図 13: “Happy Buddha” の描画結果

model	S	Alpha	Point	Pass	Time
bunny	2.7	no	362272	1	0.24
bunny	2.7	no	362272	2	0.52
bunny	2.7	yes	362272	1	0.41
bunny	2.7	yes	362272	2	0.91
buddha	1.6	no	1274573	1	0.64
buddha	1.6	no	1274573	2	1.34

表 1: IBPR の描画時間 (秒)

数 (鮮鋭化の処理をしている場合には 2)、 Time は、描画時間を示している。

4 おわりに

本稿では Image-Based Point Rendering (IBPR) による複数のレンジデータ描画方法を提案した。また、本手法での点による描画の際の隙間を埋める方法は、pull-push による方法を拡張したものとなっている。また、本手法は前処理により特別なデータ構造を生成することがなく、また毎フレーム法線を計算することにより直接レンジデータをシェーディング表示することができる。

また、本稿では、 2 パスによる描画方法を提案した。この方法により 1 パスによる描画にくらべて高品質な結果を得ることができた。この方法により描画の際に発生するぼやけを 1 パスの結果と比べて軽減することができた。更に、 IPBR においての半透明で物体を描画する手法についても提案した。

今後の課題として、今はほとんどの IBPR の実装はソフトウェアで行われているが、この処理をハードウェアによって置き換えることにより、高速な描画結果を得ることが期待される。また、このハードウェアによる実装ではプログラマブルシェーダによる高速化が考えられる。

謝辞

“Bunny” と “Happy Buddha” のレンジデータは Stanford University Computer Graphics Laboratory のご好意によって使用させていただきました。

参考文献

[1] Marc Stamminger and George Drettakis. Interactive sampling and rendering for complex and procedural geometry. In *Proc. 11th Eurographics Workshop on Rendering*, pp. 151–162, 2001.

[2] Oliver Deussen, Carsten Colditz, Marc Stamminger, and George Drettakis. Interactive visualization of complex plant ecosystems. In *Proc. IEEE Visualization 2002*, pp. 219–226, 2002.

[3] Baoquen Chen and Minh Xuan Nguyen. Pop: a hybrid point and polygon rendering system for large data. In *Proc. IEEE Visualization 2001*, pp. 45–52, 2001.

[4] Szymon Rusinkiewicz and Marc Levoy. Qsplat: a multiresoliton point rendering system for large meshs. In *Computer Graphics (Proc. SIGGRAPH 2000)*, pp. 343–352. ACM Press, New York, 2000.

[5] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: surface elements as rendering primitives. In *Computer Graphics (Proc. SIGGRAPH 2000)*, pp. 335–342. ACM Press, New York, 2000.

[6] Matthias Zwicker, Hanspeter Pfister, Jeroen van Beer, and Markus Gross. Surface splatting. In *Computer Graphics (Proc. SIGGRAPH 2001)*, pp. 371–378. ACM Press, New York, 2001.

[7] J.P. Grossman and William J. Dally. Point sample rendering. In *Proc. 9th Eurographics Workshop on Rendering*, pp. 181–192, 1998.

[8] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (Proc. SIGGRAPH 1992)*, Vol. 26, No. 2, pp. 71–78, 1992.

[9] Cass Everitt. Interactive order-independent transparency. *n*VIDIA Inc., 2000. available from http://developer.nvidia.com/view.asp?IO=Interactive_Order _Transparency.