イメージベースポイントレンダリングと色編集ツールへの応用

川田弘明* Alexandre Gouaillard‡ 森田正彦† 小檜山賢二† 金井崇*

* 慶應義塾大学 環境情報学部 † 慶應義塾大学 政策・メディア研究科

‡INSA de Lyon, France

1 はじめに

近年，3 次元測定機による計測技術の進展は著しく，3D オブジェクトの形状を表現する数万〜数十万点の 3 次元点群を高速に得ることができるようになった．一方で，測定された大容量の点群データを処理するための技術が必要とされてきた．従来，これらの点群データを後工程に利用するために，点群データからポリゴンデータを生成する [3] という手順が必要であった．しかしながら，正確なポリゴンデータを得るためにある程度の試行錯誤を伴い，その結果，点群データの取得とポリゴンデータの作成という一連のサイクルを繰り返し実行する必要があった．

ポイントレンダリングは，上記の問題を解決する一つの手段として注目されている．すなわち，点群データのみから，ポリゴンデータをレンダリングするのと同等の品質の表示結果を得ることができるため，測定機により取得した点群データを直接評価することができる．また，別の利点として，ポリゴンデータが通常保持する面のデータを必要としないため，よりコンパクトなデータ表現として定義することができる．

ポイントレンダリングは，Levoy と Whitted [6] によって初めて導入されたレンダリング手法である．点群データからポリゴンと同様のレンダリング結果を得るためには，点と点の間の隙間を埋めるための方法が必要となる．そのための一つの方法としては，一つの点に対しスプラット [8] やサーフェル [7, 10] と呼ばれる四角形や円，楕円を定義する方法が挙げられる．もう一つに，画像を利用した方法 [9, 2, 5] が提案されている．

また，シェーディングを行うためには，点群の位置情報の他に法線が定義されている必要がある．この法線は，点群から直接計算することが可能である [3] が，前処理により計算してデータとして保持する必要がある上，形状の内外を決定することができない，という問題がある．

本研究では，点群データの位置情報のみを入力とし，画像処理を利用した新しいポイントレンダリング手法について提案する．本手法では，提案する手法をイメージベースポイントレンダリング (IBPR) と呼ぶことにする．本手法の特徴として，以下の三つの点が挙げられる．

- IBPR [5] では，イメージバッファと呼ばれる，フレームバッファより解像度の低い画像を一枚用意し，その画像を利用して点と点の穴埋めを行うのが特徴である．本手法では，これに加えてスプラットによる手法を組み合わせることにより，より高品質なレンダリング結果を得ることができる．
- 主成分分析 (PCA, *principal component analysis*) を利用し，法線をフレーム毎に計算するための手法についても提案する．これより，入力データは点群の位置情報だけで済み，その結果，点群データから直接レンダリング結果を得ることができる．
- 本手法は画像処理を利用していることから，それを応用して，レンダリング結果に対して様々な効果を得ることが可能である．その一例として，点群に付加する色情報の編集ツールについて提案する．

本論文の構成は以下の通りである．まず 2 節に，本手法のもととなる IBPR について示す．3 節には，本手法における提案部分である，ハイブリッドレンダリングと主成分分析による法線計算手法につ

いて述べる．4 節には，本手法の応用である，色編集ツールについて説明する．5 節に結果を述べ，6 節に結論と展望を記す．

2　イメージベースポイントレンダリング

本節では，イメージベースポイントレンダリング [5] (以下，IBPR) について説明する．IBPR における最低限必要な入力は，点の位置情報 $\mathbf{p}_i = (x, y, z)(i = 1 \ldots n)$ のみであるが，必要に応じて色情報 $\mathbf{c}_i = (r, g, b, a)$ を付加することができる．

IBPR では，イメージバッファと呼ばれるフレームバッファより低い解像度の画像を利用してレンダリングを行う．これは，Grossman と Daily による pull-push アルゴリズム [4] に似ているが，pull-push アルゴリズムでは，複数の解像度の異なる画像を用意しなければならないのに対し，本手法では，解像度を調整することにより，1 枚の画像だけで済ませることができる．

レンダリングの際，イメージバッファの各ピクセルには，$\mathbf{p}_i$ に加えて投影後の 2 次元座標 $p_i = (p_i^x, p_i^y)$，色情報 $\mathbf{c}_i$，そして法線 $\mathbf{n}_i = (n_i^x, n_i^y, n_i^z)$ が格納される．この法線は，格納されている点群 $\mathbf{p}_i$ より計算される (3.1 節を参照のこと)．イメージバッファの解像度 w_i, h_i は，以下のようにして決定される:

$$w_i = w_s/s, \quad h_i = h_s/s \tag{1}$$

$$s = e^{-Cl/\sigma}, \quad s > 1 \tag{2}$$

ここで，w_s, h_s は，実際に描画されるフレームバッファの解像度である．l は視点から点群データまでの距離の最小値を示す．C はユーザ指定のパラメータである．また，s の値は，描画する点群データの解像度および視点からの距離により決定される．点群データの解像度 σ は次のように計算できる: すなわち，点群データ全体を $k-$ 隣接グラフとして定義し，各点に対して最も近い点との距離を計算してそれらの最小値を σ とする．$k-$ 隣接グラフの定義と近傍点の計算は視点に依存することがないため，前処理（点群データの入力時）に行うことが可能である．

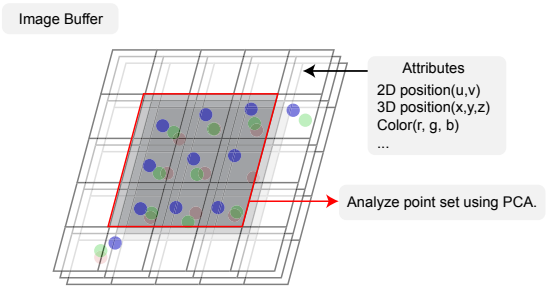


図 1: イメージバッファと主成分分析を行う範囲.

イメージバッファにこれらのデータを格納した後，データが格納されているピクセルとその隣接ピクセルから，格子上のメッシュを作成する．このメッシュを用いてシェーディング処理を行うことで，点と点の間の隙間を埋めることができる．その後，フレームバッファの大きさに戻すことでレンダリング処理が完了する．

3　イメージベースドポイントレンダリングの拡張

2 節における IBPR [5] では，特に品質に関して問題点があった．それを解決するために，本手法では主に以下の二つに対する改善を行っている．一つは，法線の計算であり，もう一つはハイブリッドレンダリングである．

3.1　主成分分析による法線の計算

ここでは，点群を格納した各ピクセルに対して，シェーディングの計算に必要な法線を計算するための手法について説明する．この法線の計算は，各ピクセルに対し主成分分析 (PCA) を行うことで得られる．主成分分析は統計的手法の一種であり，d 次元空間上の N 個の点からデータの平均，直交基底，および分散を計算することができる [1]．我々は，このうち主に主方向の決定のために用いている．

主成分分析を行う際に，各ピクセルにおける近傍の点群の位置情報が必要となる．ここでは，イメー

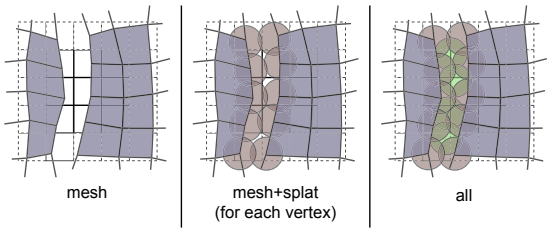


図 2: ハイブリッド・レンダリング.

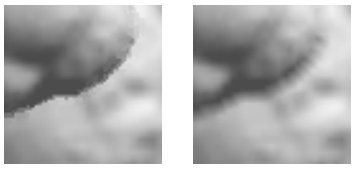
ジバッファ内の近傍のピクセルを参照することで，効率良く近傍の点群を得ることができる．図 1 には，各ピクセルに対し主成分分析を行う点群の範囲を示している．

各ピクセルとその隣接ピクセルに属する点群より共分散行列 C を計算し，この行列 C の固有値と固有ベクトルを計算する．本手法では，この固有ベクトルのうち，対応する固有値が最小のものを法線ベクトルとして割り当てている．

これらの計算はフレーム毎のレンダリングの計算の中で行われ，インタラクティブ処理が可能である．過去にも同様の手法が用いられている [4] が，計算の対象となる点群の選定に計算時間を要するため，主に前処理によって行われていた．本手法では，計算対象の点群は，イメージバッファへの格納によってあらかじめ選定されているため，高速に計算することができる．また，視点依存であることや解像度の変化にも対応できることも本手法における利点となっている．

3.2 ハイブリッドレンダリング

IBPR [5] では，利用するイメージバッファの解像度をフレームバッファより小さくすることから，最終的なレンダリング結果の際にぼけが発生する．このぼけは，主にエッジの部分に対し顕著に現れる (図 3(b))．[5] では，マルチパスレンダリングによりこの問題点を解消した．すなわち，1 パス目においてはイメージバッファの解像度をフレームバッファの解像度と等しくなるように設定し，エッジの部分を高解像度で描画してから，エッジ以外の部分を 2 パス目で描画した．



(a) (b)

図 3: レンダリング結果の比較．(a) ハイブリッドレンダリング．(b) メッシュのみのレンダリング.

本研究では，この問題を 1 パスによるレンダリングで解決するため，以下のようなハイブリッドレンダリングによる手法を新たに提案する．ハイブリッドレンダリングでは，イメージバッファにより生成される格子メッシュの他にスプラットを利用して描画を行う．このスプラットは単一色で描画されるため，描画時のぼけは解消される．このスプラットをエッジの部分に配置し，エッジ以外の部分はメッシュを利用する．ハイブリッドレンダリングの手順を図 2 に示す.

スプラットの配置において，まずエッジの検出をおこなう．エッジの検出は，イメージバッファ内の各ピクセルに格納されている法線と視点の方向の内積を計算し，90 度に近いピクセルをエッジとして選択する．スプラットの位置は次の二通りの方法によって計算する．

1. エッジとして選択されたピクセルに格納されている点群の位置の平均をとる (図 2中).
2. エッジとして選択されたピクセルに隣接するピクセルに，同様に選択されたピクセルが存在する場合，隣接する 4 つのピクセルに格納されている点群の位置の平均をとる (図 2右).

描画手順としては，まずスプラットを描き，次にメッシュを描画する．スプラットの半径は，イメージバッファの解像度時に計算した s (式 (2)) を利用している．

ハイブリッドレンダリングによる描画結果の比較を図 3(a) に示す．メッシュのみによるレンダリング結果 (図 3(b)) と比べて，エッジ部分のぼけが修正されていることが確認できる．

図 4: スタンプツール

4　色の編集手法

ここでは，IBPR を応用した色の編集ツールについて説明する．本手法では，イメージバッファを利用することで，ポイントレンダリングを画像処理として捉えることで処理を行っている．一方，色の編集ツールは，画像処理の一つとして良く行われており，例えば Adobe Photoshop などが良い例である．これらの画像処理では，主に隣接するピクセルの参照が必要となる．本手法はイメージバッファを利用していることから，これらのソフトウェアに実装されている様々な処理を模倣することが可能である．ここでは，その一つの応用例として，図 4 のようなスタンプツールの実装を行った．

2 次元画像処理におけるスタンプツールは，ある画像の一部の領域を，同じ画像内，もしくは別の画像の一領域にコピーするという機能である．この機能は，例えばノイズと考えられる部分に対し，ノイズを軽減するという目的のために使用される．

IBPR におけるスタンプツールの処理は，全ての点群の情報を，ある視点においてイメージバッファに格納した後に以下の手順で行う．

1. ユーザがコピー元となる色情報の位置を指定する．指定された位置の近傍のピクセルに格納されている点群の 2 次元座標 p_i と色 $\mathbf{c}_i$ を，あらかじめ用意しておいたバッファに格納する．

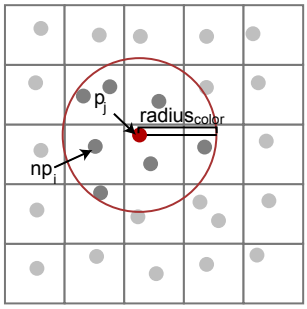


図 5: 色の対応付け

2. ユーザがコピー先となる位置を指定する．指定が行われた位置に，格納されていたバッファ内の色情報を用いて，指定された位置の近傍の点群の色情報を更新する．ここでの色の更新の様子を図 5 に示す．色を割り当てる点 p_i とその近傍の点 p_j との位置関係から，p_i の色を決定する．

p_i には，なるべく近い位置にある p_j の色を与えるため，色の影響する範囲 $radius_{color}$ を設定し，この範囲内に存在する近傍点 p_j の重み付き平均和を計算する．このときの距離は，二次元座標での距離を用いる．

$$\omega_j = \frac{|p_j - p_i|}{radius_{color}}. \tag{3}$$

p_i の色 $\mathbf{c}_i$ は以下の式により決定する．

$$\mathbf{c}_i = \frac{\sum_{j=1}^{n} \mathbf{c}_j \omega_j}{\sum_{j=1}^{n} \omega_j}. \tag{4}$$

ここで $radius_{color}$ はユーザによって決定される値であり，通常はコピーされる画像の領域の範囲を指定することで得られる．

5　結果

ここでは，本手法における結果を示す．使用した計算機環境は，CPU Pentium4 3.2 GHz，主記憶 1GB の PC である．また，本手法の結果を検証するにあたり，Stanford Bunny のレンジデータを使用している．このレンジイメージは異なる 10 方向からスキャンされたものであり，今回使用したものは

図 6: イメージベースポイントレンダリングでの "Bunny" のレンジデータの描画結果.

10 方向のレンジデータを合成したデータを利用している (362,272 点). 図 6 は本手法による Bunny のポイントレンダリング結果である. 図の中で, 一部領域を拡大した結果を図の右上に示している. 描画時間は 1.11 秒であった.

次に, 色の編集ツールの適用結果について示す. ここでは, 測定機によって得られた, 色情報が付加された Beetle のレンジデータ (559,327 点) の一部の色の欠損領域を, 今回開発したスタンプツールによって補完する, という作業を行った. 欠損部分の補完は, 近傍において似たような色を探し, その色を使って編集を行っている. 図 7(a),(c) は, 編集前の描画結果を示し, 図 7(b),(d) は編集後の描画結果を示す. 図 7(c),(d) は, 図 7 (a),(b) とは違う角度で描画したものである. 色の編集した部分は, 各図の左上もしくは左下に拡大して表示している. これらの結果より, 欠損部分が補完できていることが確認できる.

6 おわりに

本論文では, イメージベースポイントレンダリングにおいて, レンダリングの質を高めるための手法の拡張と, イメージベースポイントレンダリングを応用した色編集ツールについて提案し, 本手法が有効であることを確認した.

主成分分析による法線の生成手法においては, より鮮明な画像生成を行うことができることを確認することができた. ただ, この計算によりパフォーマンスが少々低下することも確認されており, さらなる計算時間の短縮が今後の課題として挙げられる. これには, 近年注目され多機能化されているプログラマブル・シェーダによる実装が考えられる. また, ハイブリッドレンダリングにおいては, 1 パスでの処理によって, 描画の際のぼけを軽減することができることを確認した.

色の編集に関しては, スタンプツールを一つの例として取り上げた. 編集機能に関しては, 今後は色の編集だけでなく, 点群の位置情報に対してのスタンプツールによる編集というような拡張も考えている.

謝辞

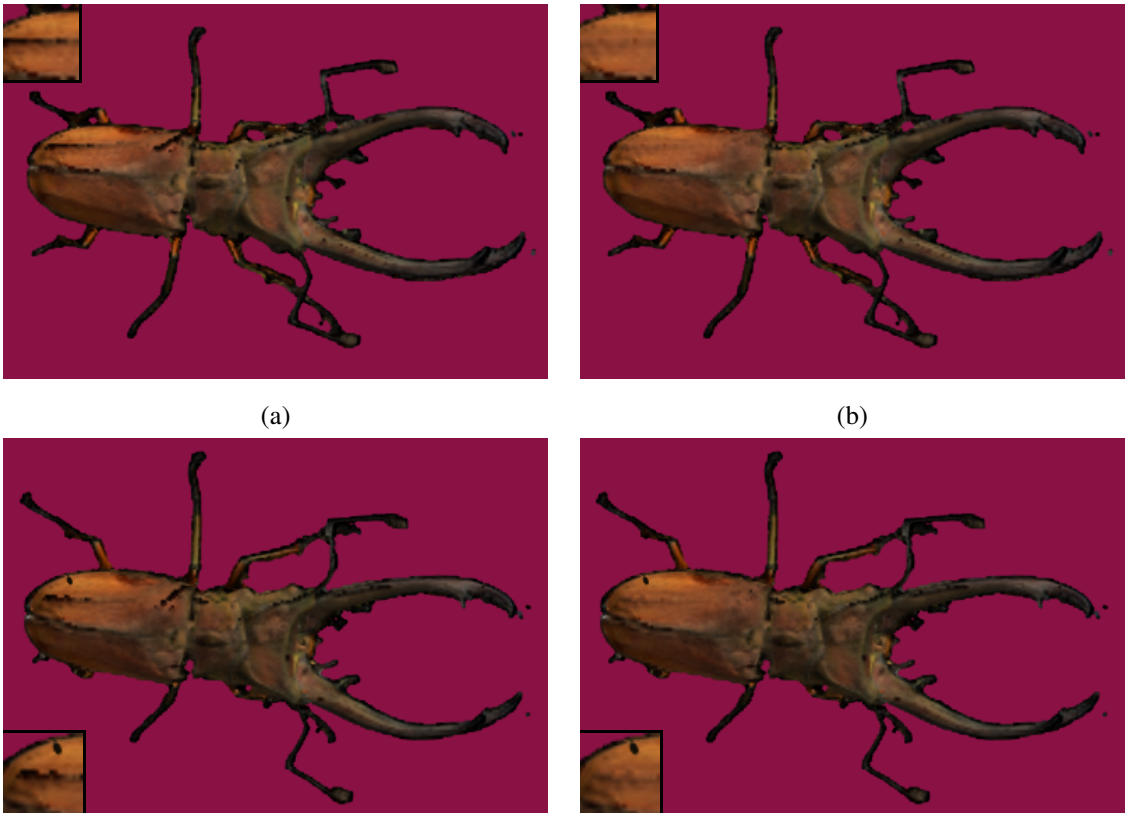
“Bunny” のレンジイメージは, Stanford University Computer Graphics Laboratory のご好意により使用させていただきました. ここに感謝の意を表します.

参考文献

[1] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann Publishers, 3rd edition, 2002.

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

[3] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Computer Graphics (Proc. SIGGRAPH 92)*, pp. 71–78. ACM Press, New York, 1992.



(a) (b)

(c) (d)

図 7: (a), (c): 色の編集前の描画結果. (b), (d): 色の編集後の描画結果.

[4] A. Kalaiah and A. Varshney. Statistical point geometry. In *Proc. Eurographics Symposium on Geometry Processing*, pp. 107–115. Eurographics Association, 2003.

[5] H. Kawata and T. Kanai. Image-based point rendering for multiple range images. In *Proc. 2nd International Conference on Information Technology & Applications (ICITA 2004)*, pp. 478–483, 2004.

[6] M. Levoy and T. Whitted. The use of points as a display primitive. Technical Report 85-022, Computer Science Department, University of North Carolina at Chapel Hill, 1985.

[7] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: surface elements as rendering primitives. In *Computer Graphics (Proc. SIGGRAPH 2000)*, pp. 335–342. ACM Press, New York, 2000.

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

[9] J. Shade, S. J. Gortler, L. W. He, and R. Szeliski. Layered depth images. In *Computer Graphics (Proc. SIGGRAPH 98)*, pp. 231–242. ACM Press, New York, 1998.

[10] M. Zwicker, H. Pfister, J. van Beer, and M. Gross. Surface splatting. In *Computer Graphics (Proc. SIGGRAPH 2001)*, pp. 371–378. ACM Press, New York, 2001.