Vol.2009-CG-136 No.4
2009/8/20

ポイントレンダリングを利用した効率的な曲面描画手法

川田弘明[†1]　大竹豊[†1]　金井崇[†1]

本研究では，ポイントレンダリングを利用した曲面の描画手法について提案する．本手法は，曲面上に点群を動的にサンプリングすることで，曲面描画をポイントレンダリングの問題に帰着させることをその特徴とするものである．メッシュにおける描画と比べて，より高品質な描画結果が期待できる．さらに，GPU による実装についても議論する．

An Efficient Method for Displaying Smooth Surfaces Based on Point-Based Rendering

Hiroaki Kawata ,[†1] Yutaka Ohtake [†1]
and Takashi Kanai[†1]

In this paper we describe a method for displaying smooth surfaces using point-based rendering approach. The main characteristic of our method is that points are sampled dynamically during rendering process. It is expected that our approach achieves high-quality rendering results compared to mesh-based methods. In addition, we discuss the implementation of our method on GPUs.

1. はじめに

近年，複雑な形状を表現するために陰関数曲面形式が利用されている．通常，陰関数曲面を表示する際には，マーチングキューブ法[3] などの等値面生成アルゴリズムによるメッシュへの近似や，レイキャスティング法などが用いられる．

本研究では，陰関数曲面形式の一つである SLIM (Sparse Low-degree Implicit) 曲面[4] に対し，点のサンプリングを基本とした新しい描画手法を提案する．既存の描画手法として，GPU を用いたレイキャスティング法による手法[1] が挙げられる．この手法では，プログラマブルシェーダの中で各ピクセルからのレイと多項式との交点計算を行う必要がある．本手法は，描画手法の過程において，曲面とレイとの交点計算を必要としないため，次数の高い多項式においても描画を行うことができる．また，折り目や角をもつ場合においても，アルゴリズムを変えることなく描画することができる．

本研究では，ポイントレンダリングによる描画を基本とする．メッシュを描画する際には頂点間の接続情報を必要とするが，ポイントレンダリングではこのような情報を必要とすることなく，点の座標のみを利用して描画を行うことができる．このような特徴から，点のサンプリングにより描画する点を増加しても，接続情報を構築する必要がないため，効率よく描画処理を行うことができる．

過去に同様の研究が行われている．Levoy らは，点群のサンプリングを行い描画する手法が提案されている[2]．この方法は，様々な形式で表現された形状に対して，点群のサンプリングを行い，描画の際の基本的なプリミティブとして点を描画する手法である．点の描画を行う際には，テクスチャフィルタリングを行い，点と点の間の穴を埋める処理を行っている．本手法は，この手法を基本としているが，サンプリングされた点群の描画においてテクスチャフィルタリングを行わずにサンプリングをより細かく行うことで高品質な描画結果を得ることを目指している．

ポイントレンダリングは，スプラットによる高速な描画手法である Qsplat[6] が提案されている．Qsplat では，四角形や円のような形状を点の代わりに描画する．これは，スクリーン空間における解像度が足りない場合に，適切な半径を設定することにより点と点との隙間を埋めるためである．スプラットによる穴の隙間を埋める処理は，高速ではあるが高品質な結果を得ることが困難である．このため，本手法ではスプラットによる描画は行わず，点の入っているピクセルを塗りつぶすようにする．

また，テクスチャフィルタリングを使用した高品質な描画手法[7][5] が提案されている．特にそれらの手法の一つである Surface Splatting では，EWA フィルタを点群を描画する際に使用することで高品質な描画結果を得ることができる．一方で，本手法ではテクスチャフィルタリングを使わずに高品質な描画結果を得ることを目的としている．

2. 関数曲面形式の SLIM 曲面

本研究では，関数形式の関数を基本とした SLIM 曲面の描画を行う．SLIM 曲面は中心位

†1 東京大学
The University of Tokyo

Vol.2009-CG-136 No.4
2009/8/20

置と半径を持つサポート球の集まりで構成され，各のサポート球には一つ以上の多項式関数が定義される．サポート球は，位置 $\mathbf{p} = (cx, cy, cz)$，半径 r，基底変換の回転行列 $\mathbf{M}$，多項式の関数の情報を持つ．サポート球の持つ関数は，2 変数多項式関数であり，2 次式の場合以下のように表される．

$$w = f(u, v) = Au^2 + Buv + Cv^2 + Du + Ev + F \quad (1)$$

本手法では，関数 $f(u, v)$ を陰関数の形式に変換を行わず値を直接計算する．関数 $f(u, v)$ は，サポート球の持つローカル座標系により表現されている．このため，曲面上の点の位置を求めるときは，ローカル座標系での座標値を回転行列によりグローバル座標系での座標値に変換してから重み付き和を計算する必要がある．ローカル座標系での座標値を $\mathbf{u}_i = (u_i, v_i, w_i)$ とすると，曲面上の座標値は $\mathbf{x} = \sum \omega_i(\mathbf{M}_i\mathbf{u}_i + \mathbf{p}_i)/\sum \omega_i$ のように表すことができる．重み ω_i は，サポート球の中心に近いほど影響が強くなるような関数を利用している．

3. 関数曲面形式の SLIM 曲面の描画

本研究では，点群を直接サンプリングすることにより関数曲面の描画を行う．サンプリング手法に関しては 4 節で詳しく説明する．

SLIM 曲面においては，各サポート球において他のサポート球と重なっている箇所に対して重み付平均和を計算する必要がある．このため，3 パスで描画を行う．描画処理の手順は次のようになる (図 1).

- 1 パス目では，サンプリングされた各点をスクリーンに投影する．投影された点の座標に対応する画像のピクセルに点のデプス値のみを記録する．
- 2 パス目では，サンプリングされた各点をスクリーンに投影する．投影された点のデプス値と対応するピクセルにおける 1 パス目で格納したデプス値の比較を行い，一定の閾値 t の距離の範囲にある場合にシェーディングの計算を行う．計算された色値を重み ω で重み付けして加算する．また重みも別に加算する．
- 3 パス目では，2 パス目で計算した各ピクセルの色値を重みの和で割り算する．

閾値 t は，ここではサポート球の半径 r を利用している．2 パス目では，シェーディングの計算を行うが，ここで必要となる法線ベクトルは $(-\partial f/\partial u, -\partial f/\partial v, 1)$ のように計算している．また，投影は平行投影を行っているものとする．

4. 点群のサンプリング

点群のサンプリングは，関数 $f(u, v)$ の値を計算することによって行う．u,v の値をそれぞれ $[-r, r]$ の範囲において $-r$ からサンプリングの間隔 n を加えて r になるまで値を変化させることにより点のサンプリングを行う．

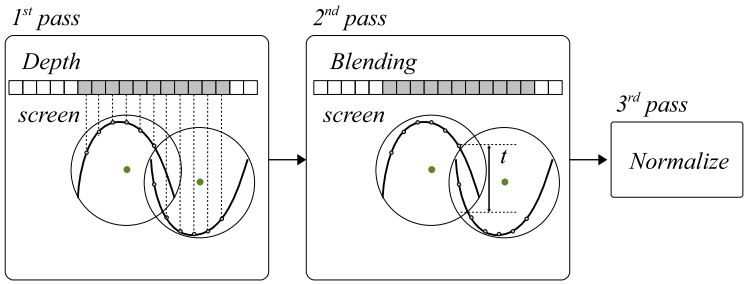


図 1 本手法における描画処理

ここで，サンプリングを行う間隔 n は，サポート球をスクリーンに投影することにより，サポート球が影響する範囲の幅の長さから計算する（図 3）．投影されたサポート球の幅を N とすると，n は，$S(r/N)$ により計算することができる．平行投影の場合には，$N = 2r$ とすることができる．S を 2 とするとサポート球をビルボードとして描画した時のスクリーン空間におけるピクセルの幅とほぼ一致する．

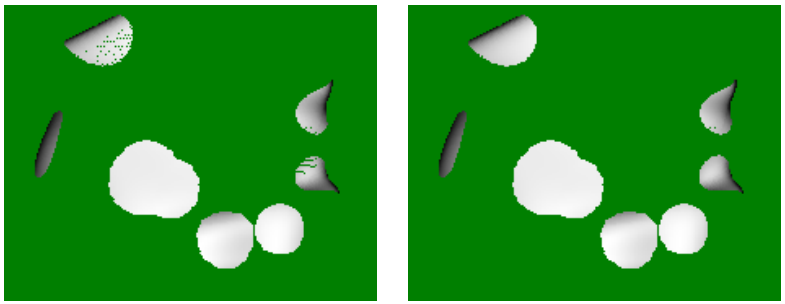
しかし，一定間隔でサンプリングを行った場合には，視点と関数の勾配が大きい箇所の角度が 90 度に近い場合にサンプリング点が少なく穴が空くという問題がある（図 4 左）．このため 2 段階目のサンプリングを行い穴を埋める処理をする．

4.1 2 段階目のサンプリング

サポート球内における等間隔のサンプリングでは，穴が発生する場合がある．このような問題に対応するため，一定の間隔でサンプリングを行っている処理の過程で，穴の空いている箇所を動的にサンプリングするという処理を行う（図 4 右）．この処理においては，スクリーン空間に投影された 2 点の範囲から動的サンプリングが必要がどうかを判定する．

判定の処理を説明する．はじめに，サンプリングを行う点 $f(u, v)$ と $f(u + n, v + n)$ をスクリーンに投影する．それぞれの点の間のピクセル数が 1 以上の値であれば穴があいていると判定する．動的サンプリングを行う際には，この穴の空いているピクセル数に応じてサンプリング点の数を決定する．

図 2 に動的サンプリングの結果を示す．(a) は等間隔にサンプリングを行った場合の結果



(a) (b)

図 2 2 段階目のサンプリングによる穴埋めの処理

で，関数曲面の一部に穴があいているのが見て取れる．(b) は，2 段階目のサンプリングを行った結果であり，穴が埋められているのがわかる．

5. 折り目の表現と描画

SLIM 曲面では折り目を表現することが可能である．この場合，該当する箇所では，サポート球が 2 つ以上の多項式関数を持つ．

折り目と判定されたサポート球は，関数の評価方法が異なるのみで，描画処理，サンプリング処理の方法は同じである．関数の評価を行う際，2 つの多項式関数のうちどちらかの関数の最大値，または最小値を選択する．図 5 は，(a) が最小値，(b) が最大値をそれぞれ選択した場合を示している．最小値と最大値の選択に関する情報は，SLIM 曲面に変換を行う際にあらかじめ決定され与えられている．

6. 結果と議論

本手法による結果を示す．結果は，512 × 512 ピクセルの解像度で計算しており，Core 2 Quad 3GHz の CPU の計算機を使用した．

図 6 に，本手法によるいくつかのモデルに対する描画結果を示す．また，表 1 には，モデルのノード数，スケール，描画時間，および，描画の際の点の数を示す．これより，描画時間は，ノード数ではなく，主にサンプリングされた点の数に影響されていることがわかる．サンプリングされる点の数は N によって決定されるため，この値が大きい場合には描画速度が低下すると考えられる．また，2 段階目におけるサンプリングにより増加する点が多ければ多いほど，描画速度が低下するものと考えられる．

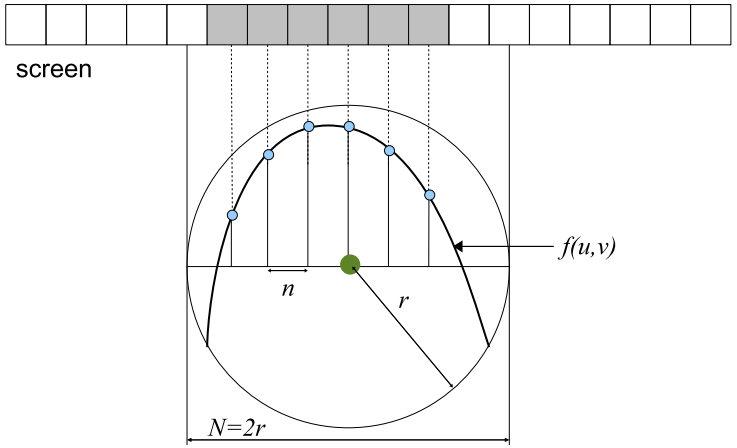


図 3 サポート球の範囲におけるサンプリング

また，図 7 は，2 次多項式の関数表現による折り目を表現した fandisk（3829 ノード）の描画結果である．描画時間は，0.983 秒であった．エッジの一部で欠けている箇所が見られるが，これはサンプリング不足による影響であると考えられるが，詳しい検証と改良は今後の課題とする．

6.1 GPU による処理の議論

本研究による方法は，GPU による処理が可能であると考えている．GPU ではプログラマブルシェーダを利用することで，描画処理をプログラムによって制御が可能である．ここでは，この機能を使用して GPU において処理する方法を議論する．

本研究における描画処理は，大きく分けて二つの処理に分けられる．一つ目は，サンプリングを行う処理，二つ目はサンプリングされた点の重み付平均和を計算する処理である．

一つ目の点のサンプリング処理は，ジオメトリシェーダによって処理を行う手法が考えられる．ジオメトリシェーダは，入力された頂点に対して，頂点数を増やすことが可能なシェーダーである．頂点シェーダは，入力された頂点に対しての処理であるが，ジオメトリ

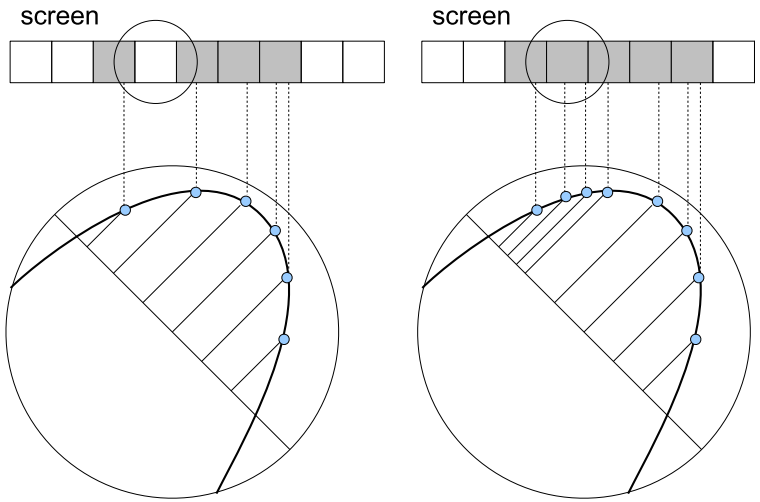


図 4 点の動的サンプリング

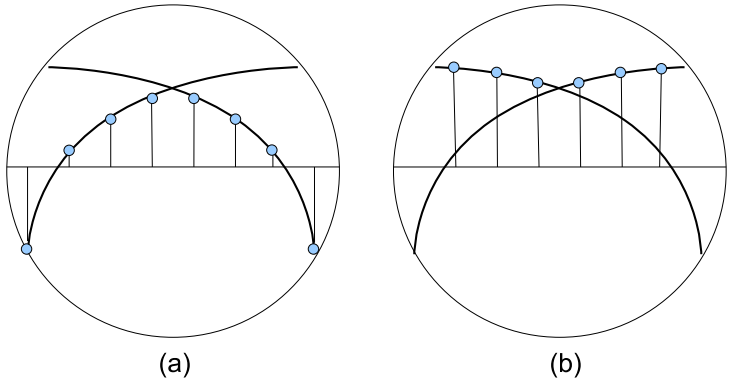


図 5 折り目における点のサンプリング

name	nodes	scale	time	points
moai	855	1.0	0.904	1,550,228
moai	855	0.5	0.218	375,421
monk	20811	1.0	0.593	766,135
dino	2653	1.0	0.405	687,099

表 1 各モデルのノード数，スケール，描画時間，描画の際の点の数

シェーダでは入力された頂点に加えて，新たな頂点を GPU において加えることが可能であるところが異なる．この機能を使用することにより，点群のサンプリング，そしてサンプリングされた点の座標変換を GPU において効率よく処理することができると考えている．

二つ目の重み付平均和を計算する処理については，これまでに提案された GPU によるポイントレンダリングの描画処理の手法を利用することが可能であると考えている．この処理は，Kanai らによる GPU による SLIM の描画手法[1] にも使用されている．この処理は 3 パスで行われるが，本手法においても基本的には同じ処理を行うことから同じ手法が利用できると考えている．

今後は，ここで議論を行った GPU による処理を実装することにより，描画の処理速度を速くできることを期待している．

7. おわりに

本研究では，点群サンプリングによる SLIM の描画手法を提案した．点の動的サンプリングにより，滑らかな曲面を描画できることを示した．また，折り目を持つ曲面に関しても，アルゴリズムを変えることなく描画できることを合わせて示した．

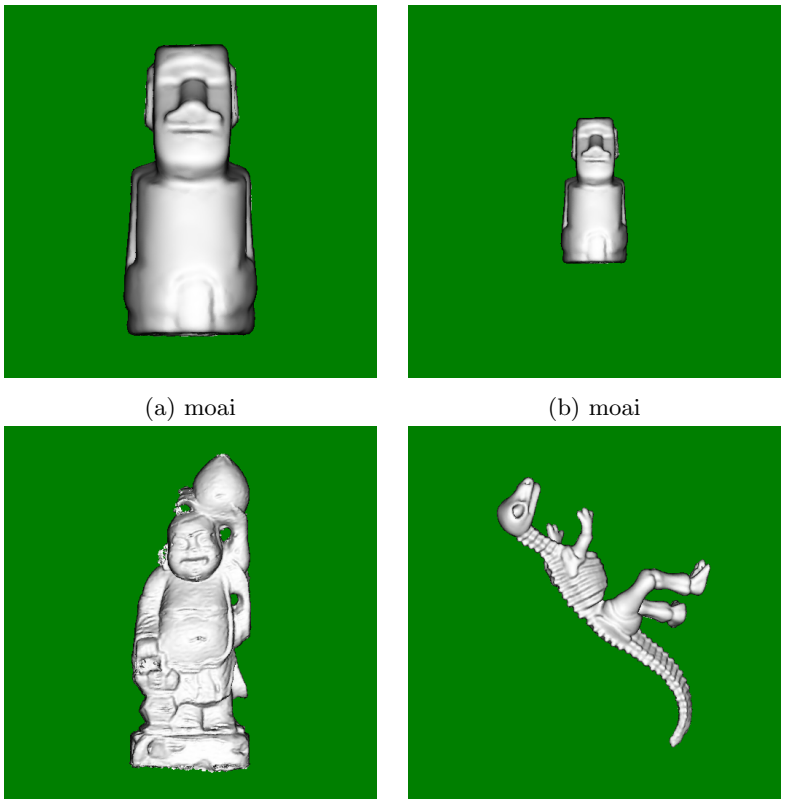
しかし，高速な描画速度を得るにはサンプリングの手法を改良する必要があると考えられ，今後はこれらの改良を行うことを考えている．また，点群のサンプリングにおける問題点として，各サポート球のサンプリングされた点がピクセルに情報を格納する際に，格納される点の数が均一でないということが挙げられる．今後は，このような問題に対しても改良を行うことを考えている．

また，今後は GPU による実装を行い描画速度の検証を行う．

参 考 文 献

1) T. Kanai, Y. Ohtake, H. Kawata, and K. Kase. GPU-based rendering of sparse low-degree implicit surfaces. In *Proc. GRAPHITE '06*, pp. 165–171, 2006.
2) M. Levoy and T. Whitted. The use of points as a display primitive. In *Techni-*

Vol.2009-CG-136 No.4
2009/8/20



(a) moai (b) moai

(c) monk (d) dino

図 6 描画結果

(a) (b)

図 7 fandisk の描画結果

cal Report 85-022, Computer Science Department, University of North Carolina at Chapel Hill, 1985.

3) W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987.

4) Y.Ohtake, A.Belyaev, and M.Alexa. Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. In *Proc. Symposium of Geometric Processing 2005*, pp. 149–158, 2005.

5) H.Pfister, M.Zwicker, J.van Baar, and M.Gross. Surfels: surface elements as rendering primitives. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 335–342, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

6) S.Rusinkiewicz and M.Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 343–352, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

7) M.Zwicker, H.Pfister, J.van Baar, and M.Gross. Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 371–378, New York, NY, USA, 2001. ACM.