曲面上の点群データの確率過程的並列リサンプリングとその可視化への応用

田中 覚，仲田 晋，木村 彰徳
理工学部 情報学科

1 はじめに

レーザによる3次元スキャナの発達に伴い，設計図などが存在しない歴史的・文化的建造物に対して，現物測定によって，3次元コンピュータ・グラフィックスのための形状モデリングを行う機会が増えている．

しかし，レーザ光を掃引して得られる対称物上の点群データに関しては，その品質に改善すべき点も多い．例えば，観測点から見て輪郭線に近い部分や凹凸の激しい部分には，充分な数の点群が得られにくい．このような部分的な点群データの不足は，複数の観測点を設定することである程度補うことが出来るが，多くの場合充分ではない．また，測定時のノイズなどのために，異なる観測点で得られたデータから整合性のあるポリゴン・メッシュを生成するのには，しばしば多大な労力を要する．結局，複雑な物体の計測の際には，手作業を含む多くのデータ処理作業が要求されることになる．

本研究では，上記の現状を改善することを最終目標とし，複雑な曲面上に与えられた点群データの品質を改善する手法を提案する．本手法では，与えられた点群データを非線形補間して陰関数曲面化し，これを確率過程的アルゴリズムを用いた並列計算によって高速リサンプリングして，曲面上の一様かつ高密度な補間点群を新たに生成する．リサンプリングの並列効率は非常に優れており，CPU数にほぼ正比例して計算速度が向上する．また，得られた高密度な補間点群を用いて，対象曲面をポリゴン近似を用いずに精密に可視化できる（図1, 2）．

以下では，入力データとして曲面上に与えられる点を「参照点」，リサンプリングの出力データとして曲面上に生成される点を「補間点」と呼んで区別する．

2 参照点群の陰関数曲面化

参照点群を非線形補間して陰関数曲面化するのには，Turk-O'Brien の方法 [1] を用いることができる．基底関数には以下のものを選択する：

$$f_i(\mathbf{q}) = |\mathbf{q} - \mathbf{P}_i|^2 \ln |\mathbf{q} - \mathbf{P}_i| \,. \qquad (1)$$

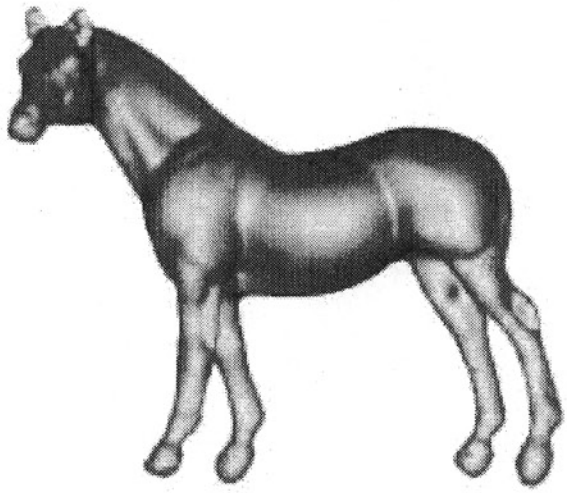

図 1: 補間点群を用いた粒子レンダリングの例1．

ここで，$\mathbf{q} = (q_1, q_2, q_3)$ は3次元座標，$\mathbf{P}_i$ は i 番目の参照点の位置ベクトルである．この基底関数の適当な線形結合を作り，それをゼロと置くことで，参照点群を通る陰関数曲面が得られる．

3 確率過程サンプリング法

前節で述べた陰関数曲面化は，参照点群を，ポリゴン化のような線形補間でなく，より精密に非線形補間するものと言える．非線形補間の結果を正確に可視化に反映させるには，得られた陰関数曲面をリサンプリングし，曲面上の全領域にわたって，出来るだけ高密度な補間点群を生成すれば良い．

補間点群の生成には，最近我々が開発した「確率過程サンプリング法」[2, 3, 4] を用いる．方程式

$$F(\mathbf{q}) = 0 \,, \qquad (2)$$

で定義される陰関数曲面を I_F とする．ここで，$F(\mathbf{q})$ はスカラー関数である．以下の確率微分方程式を差分化

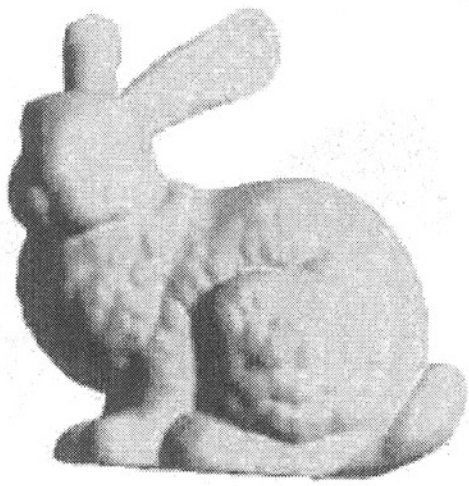

図 2: 補間点群を用いた粒子レンダリングの例 2 .

して数値的に解くことで，I_F 上に一様なサンプル点群を高速生成することが出来る:

$$dq_i(t) = dq_i^{(\mathrm{T})}(t) + dq_i^{(\mathrm{S})}(t) + dq_i^{(\mathrm{N})}(t) \ . \qquad (3)$$

ここで，t は仮想的に導入された時間変数であり，$dq_i^{(\mathrm{T})}$, $dq_i^{(\mathrm{S})}$, $dq_i^{(\mathrm{N})}$ は，それぞれ，

$$dq_i^{(\mathrm{T})} \equiv \sum_{j=1}^{d} P_{ij} dw_j \ , \qquad (4)$$

$$dq_i^{(\mathrm{S})} \equiv -\frac{\alpha}{|\nabla F|^2}\left(\frac{\partial F}{\partial q_i}\right) \mathrm{Tr}\left\{(\partial^2 F)\cdot P\right\} dt, \qquad (5)$$

$$dq_i^{(\mathrm{N})} \equiv -K\alpha\left(\frac{\partial F}{\partial q_i}\right)\frac{F}{|\nabla F|^2} dt \ , \qquad (6)$$

で定義される．ランダムさを生成するのは，ガウス型のランダム変数 $dw_i(t)$ であり，これは以下の統計的性質を満足する：

$$\langle dw_i(t)\rangle = 0 \ , \qquad \langle dw_i(t) dw_j(t)\rangle = 2\alpha\delta_{ij} dt \ . \qquad (7)$$

$\langle\cdot\rangle$ は統計平均を表す期待値記号であり，α は正の定数，δ_{ij} はクロネッカーのデルタである．$dw_i(t)$ に作用している P_{ij} は I_F 上への射影行列であり，次式で定義される．

$$P_{ij} \equiv \delta_{ij} - \frac{1}{|\nabla F(\mathbf{q})|^2}\frac{\partial F}{\partial q_i}\frac{\partial F}{\partial q_j} \ . \qquad (8)$$

式 (5) の中の $(\partial^2 F)$ は，要素が $\partial^2 F/\partial q_i \partial q_j$ で与えられる行列である．

確率微分方程式 (3) が生成する確率過程は，陰関数曲面 I_F 上に閉じ込められたウィーナー過程となる．これは，直感的には，曲面上に閉じ込められた微小な 1 粒子のブラウン運動（不規則運動）に対応する（図 3）．ブラウン運動の軌跡上の点を集めれば，I_F 上の一様な補間点群が得られる．確率微分方程式 (3) を数値的に解くことを長く続ければ続ける程，多くの補間点群が得られる．そして，計算時間は軌跡の長さ，すなわち得られる補間点数に正比例する．このため，大量の補間点群を生成しても，計算時間はそれほど急速に増加しない．これが，確率過程サンプリング法の高速性の理由であり，また，大量の補間点群の生成に適している理由でもある．

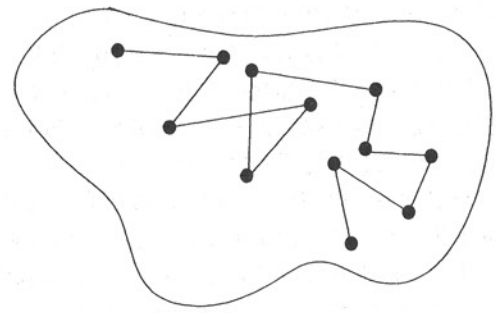

図 3: 曲面上のブラウン運動．小さな円は補間点を表す．

4 並列化

4.1 確率過程サンプリング法の並列化

Turk-O'Bien の方法で生成した陰関数曲面を記述する方程式は，参照点の数と同程度の非常に多数の項を持つ．これをリサンプリングするには，どのようなアルゴリズムを用いるにせよ，長い計算時間を要する．確率過程サンプリング法の場合には，その実行速度は項数に反比例して低下する．しかし，確率過程サンプリング法は，並列処理によって大幅な高速化が可能である．したがって，PC クラスタなどのマルチ CPU 環境では高速実行が可能である．

確率過程サンプリング法の並列化は極めて簡単である．通常の確率過程サンプリング法は，前節で述べたように，1粒子のブラウン運動に対応する．これを，図4のように，独立なブラウン運動を行う多粒子系に置き換え，各粒子が生成する補間点群を単純に重ね合わればよい．各粒子間には相互作用は全く無いので，計算の完全並列化が可能である．ひとつの粒子のブラウン運動をひとつのCPUに生成させれば，CPUに比例した高速化が可能になる．

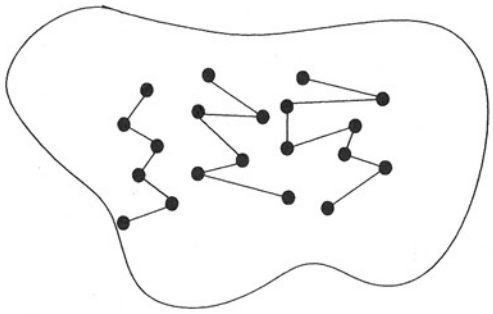

図 4: 多粒子系のブラウン運動による補間点群の並列生成.

並列計算においては，ひとつの管理プロセスを設定し，残りのプロセスを，リサンプリングを実行する計算プロセスとする．各プロセスは別々のCPUで実行される．管理プロセスは，リサンプリングの開始前に，各計算プロセスに別々の初期位置（$\mathbf{q}(t=0)$）と乱数シードを分配する．これにより，各計算プロセスごとに独立なブラウン運動が割り当てられる．リサンプリングが開始されると，各計算プロセスは互いに独立に補間点群を生成し，それらを管理プロセスに転送する．管理プロセスは，受け取った補間点群を統合してファイルに保存するか，またはネットワークに接続されたグラフィックス用のPCに転送する．前者の場合はオフライン，後者の場合はオンラインで可視化が行われる．

4.2 数値実験

我々は，図5のような馬の形状を用いて，リサンプリングの数値実験を行った．図5に描かれた7000点の参照点データを陰関数曲面化したものを用いてリサンプリングを行い，計算速度とCPU数（プロセス数）の関係を調べた．結果を図6に示す．

図6に示すグラフの中の四角い点は，いろいろな並列度において，計算プロセスで補間点群と法線ベクトル[1]を計算し，管理プロセスに集めるまでの時間を計測したものである．一方，丸い点は，補間点群の位置ベクトルのみを計算して集めた場合の結果である．

図6より，理論どおり，CPU数に比例して計算速度が増大していることがわかる．図中には，データ点を通る直線も最小2乗法で求めて描き加えてある．p_0, p_1は，それぞれ，直線の縦軸切片と傾きである．グラフが原点を通らないのは，リサンプリングを行わない管理ノードがひとつ存在するためである．実際，横軸切片$-p_0/p_1$はおよそ1となっている．

図7にリサンプリングによって得られた補間点群（10万点）を示す．一様かつ高密度な補間点群が得られていることがわかる．図5の参照点群には局所的に低密度な箇所が多数あるが，図7の補間点群ではそれらの箇所がきれいに埋め尽くされている．

なお，図7で馬の足の部分が丸くなっているが，これは足の裏に参照点がほとんど無いためと思われる．

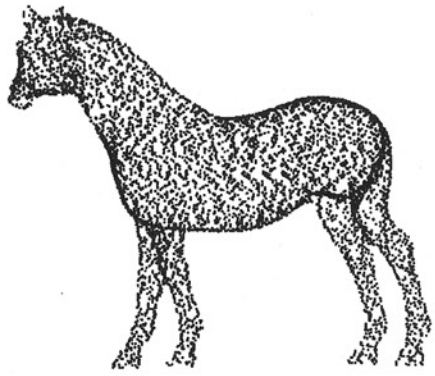

図 5: 数値実験で使用した参照点群 (7,000 点).

5 補間点群を用いた可視化

確率過程サンプリング法を用いれば，一様かつ高密度な補間点群（及び各補間点における法線ベクトル）を高速に生成することが出来る．これらの補間点のそれぞれに，その点での陰関数曲面の反射光の色を割り当てて，微小な球として可視化すれば，陰関数曲面のレンダリングが可能となる．我々は，これを「粒子レンダリング」[2] と呼んでいる．粒子レンダリングには，次の2つの意味がある．

[1]法線ベクトルは補間点における$F(\mathbf{q})$の勾配から計算できる．補間点を陰影をつけて可視化するのに用いられる．

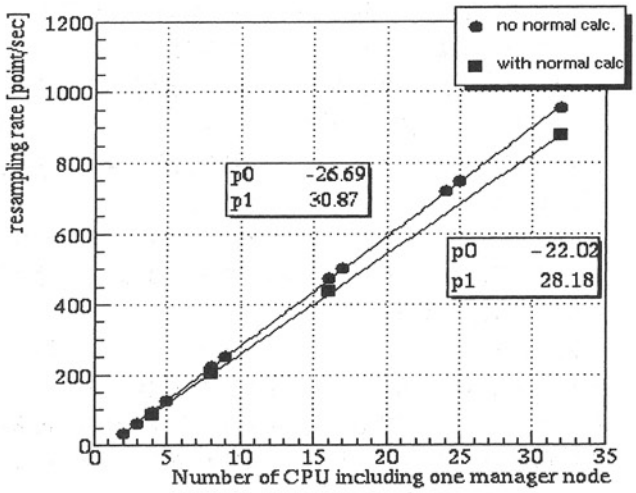

図 6: CPU 数と計算速度の線形関係．p_0, p_1 は，それぞれ，最小 2 乗法で求めた直線の縦軸切片と傾きをあらわす．

1. ポリゴン化に要する時間を節約できる．
2. 参照点群を非線形補間した陰関数曲面を，ポリゴン化という線形近似を経ずにそのまま可視化できる．

図 1，2 に，確率過程サンプリング法によるリサンプリングによって得られた補間点群を用いて粒子レンダリングした例を示す．このような滑らかなレンダリングが可能なのは，陰関数曲面化によって滑らかに変化する法線ベクトルが得られ，また，確率過程サンプリング法によって高密度の補間点群が得られるからである．

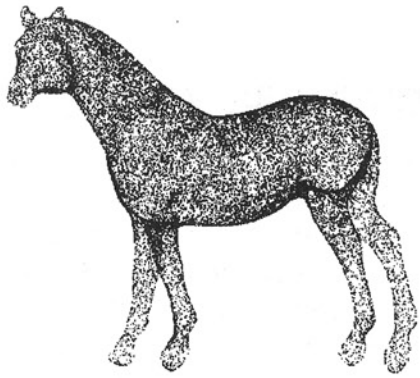

図 7: 生成された補間点群（100,000 点）．

6 おわりに

本報告では，「曲面上の参照点群データ ⇒ 陰関数曲面化 ⇒確率過程サンプリング法による補間点群の生成⇒粒子レンダリング」という，曲面の新しい可視化手法を提示した．今後はこの手法を，京都の歴史的・文化的建造物や，その周辺に存在する芸術作品等に適用していく予定である．具体的には，祇園祭の山鉾や，日本庭園の庭石などへの適用を考えている．

陰関数曲面化と確率過程サンプリング法の組み合わせには，曲面の可視化以外にも，曲面の衝突判定や衝突解析 [2, 5]，表面積の計算など，色々な応用が考えられる．これらの技術を京都の歴史的・文化的建造物などに適用する計画も進行中である．

参考文献

[1] G. Turk, J. F. O'Brien, "Modeling with implicit surfaces that interporate," ACM Transactions on Graphics, 21(4), (2002), 855-873.

[2] S. Tanaka, A. Shibata, H. Yamamoto, H. Kotsuru, " Generalized Stochastic Sampling Method for Visualization and Investigation of Implicit Surfaces," Computer Graphics Forum 19(3), (2001), 359-367. (Proceedings of Eurographics 2001).

[3] S. Tanaka, A. Morisaki, S. Nakata, Y. Fukuda, H. Yamamoto, "Sampling Implicit Surfaces Based on Stochastic Differential Equations with Converging Constraint," Computers & Graphics, 24(3), (2000), 419–431.

[4] S. Tanaka, T. Nakamura, M. Ueda, H. Yamamoto, K. Shino, "Application of the Stochastic Sampling Method to Various Implicit Surfaces," Computers & Graphics, 25(3), (2001), 441-448.

[5] S. Tanaka, Y. Fukuda, H. Yamamoto, "Stochastic Algorithm for Detecting Intersection of Implicit Surfaces," Computers & Graphics, 24(4), (2000), 523-528.