陰関数曲面の非多様体への拡張，及びその高速な可視化

山崎 俊太郎* 加瀬 究† 池内 克史‡

概 要

ボクセルサンプリングされた関数による陰関数曲面は多様体形状しか扱うことが出来ない. 本報では，関数の不連続性を考慮することで非多様体形状を陰関数表現する手法を提案する．また，グラフィクスハードウェアを利用して陰関数表現された表面形状を効率的に描画方法を提案する.

Representation and visualization of non-manifold implicit surfaces

Shuntaro Yamazaki* Kiwamu Kase† Katsushi Ikeuchi‡

Abstract

Implicit surfaces defined by a voxel-sampled implicit field cannot represent non-manifold shapes. We present the method of representing non-manifold surfaces in this form taking account of the discontinuity of both implicit functions and interpolating functions. We also present an effective method of visualizing implicit surfaces making the best of a modern graphics hardware.

1 はじめに

コンピュータグラフィクスで物体の表面形状を記述する際に用いられる表現形式は，パラメトリック表現と陰関数表現に分類することが出来る．本報では, 非多様体曲面をパラメトリック表現から陰関数表現へ変換する手法を提案する. またそのための具体的なアルゴリズムとして, 三角形メッシュを入力としてボリューム化された陰関数場を出力する手法について説明する. また，このようにして得られた陰関数場に対してボリュームレンダリングの手法を適用し，非多様体曲面を直接描画する手法に関しても提案する. この方法はグラフィクスカードのプログラム可能なテクスチャ合成機能を利用して効率的に実装することができ, 対話的な速度で高画質の曲面を表示できる.

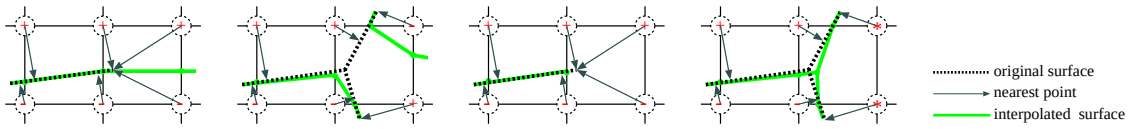
2 非多様体曲面の陰関数表現

2.1 陰関数曲面

3 次元空間内の曲面 $S \subset \mathbf{R}^3$ に対して, 次を満たす実数値関数 $f(p)$ を考える.

$$p \in S \quad \Leftrightarrow \quad f(p) = 0. \tag{1}$$

*東京大学 (University of Tokyo), 理化学研究所 (RIKEN)
†理化学研究所 (RIKEN)
‡東京大学 (University of Tokyo)



(a-1) (a-2) (b-1) (b-2)

図 1: 陰関数表現された非多様体形状. (a-1) (a-2): 従来法. (b-1) (b-2): 提案手法.

ただし $p \in \mathbf{R}^3$ は 3 次元空間内の点である. f は S への距離に応じた値を返す関数と考えることができ, S は f の値が 0 の等値面である. 曲面 S の式 (1) 右辺の形式による定義を陰関数表現と呼び, このとき S を陰関数曲面と呼ぶ.

実数値関数 f で定義される関数場を計算機で処理する時には, f をサンプリングする必要がある. そこで, あらかじめ f 格子点でサンプリングしてボリュームデータの形で保持することにより, 陰関数場に対する処理を効率的に行うと都合が良い. 本報ではこのボリュームデータを陰関数ボリュームと呼び, 陰関数場はボリュームで定義されるものとする. 陰関数場がボリュームで与えられる場合, 格子点の間の点における陰関数の値はサンプル点での値を補間して決定する.

2.2 非多様体曲面

3 次元空間で, 集合 S が 2-多様体であるとは, S 上の任意の点における無限に小さい近傍が 2 次元の円盤と位相同相であることをいう. 以降では, 多様体という言葉を用いた場合, 2- 多様体を指すものとする. 陰関数場が, サンプリングされた値の線形補間で与えられる場合には, 空間内の任意の点で関数値が微分可能である. 従って等値面上の任意の点 p の近傍は法線が ∇f の微小平面近似することができ, このとき陰関数曲面は多様体である.

一方, S が非多様体であるとは, 表面上のある点の近傍領域において, 座標軸が 1 つ以下しかとれないか, もしくは 2 次元局所座標系のとり方が 3 通り以上あることをいう. 本報で扱う形状はパラメトリック曲面, 特に三角形メッシュで表現できる曲面であり, 個々の三角形は自己交差を考えない限り, 境界以外は多様体である. 従って曲面上で多様体の性質を満たさなくなる点が存在するのは三角形の境界のみであり, これは次の 2 つの場合に分類できる.

表面に境界がある場合: ある三角形の境界上の点が, 別のどの三角形の境界上にもない場合. 2 次元で模式化された例を図 1 (a-1) に示す.

表面に分岐がある場合: ある三角形の境界上の点が, 別の 2 つ以上の異なる三角形の境界上にある場合. 2 次元で模式化された例を図 1(a-2) に示す.

ただし入力の三角形の集合内の異なる 2 つの三角形が, その境界以外の点を共有する場合には, 共有している点を含む直線に沿ってこれらの三角形を分割することにより, 三角形同士の交差は生じないものとする.

図 1(a-1)(a-2) にある通り, 従来手法を用いてこれらの非多様体形状を陰関数曲面に変換すると形状が大きく変わってしまう. そこで, 非多様体曲面を正しく陰関数表現するためには, これらの特徴を陰関数場の中で保持し, また表面として再構成するアルゴリズムを構築する必要がある.

2.3 距離関数と補間関数の設計

第 2.1 節で述べたように, 陰関数ボリュームを用いて曲面を陰関数表現するためには, 陰関数場を構成するための距離関数と, 表面の位置を決定するための補間関数を定義する必要がある. そこでこれらをうまく定めることにより, 第 2.2 節で挙げた 2 つの非多様体の特長を陰関数曲面の枠組みの中で表現する方法を考える.

まず曲面に境界がある場合に関して, 従来の陰関数表現では境界部分が延長されて, もともと存在しなかった表面が過剰に生成されてしまう (図 1(a-1)). これは, 従来の陰関数を用いて定義された陰関数ボリューム中にも正しい表面の位置が定義されているが, 補間によって不要な面が生成されていることを意味する. そこで元の曲面が存在しない位置には表面を生成しないように補間関数を拡張することで, 表面の境界を表現することが可能である (図 1(b-1)).

距離関数として符号付距離を用いた場合には, 補間関数として線形補間を用いて表面の位置を決定していた. そこでこれを拡張し, 補間の対象となる 2 つの互いに隣接するボクセルでの符号付距離の差が一定幅より大きい場合には, このボクセル間には表面を生成しないようにする. 即ち任意の 2 つのサンプル点の間隔を w, 値をそれぞれ u, v $(u \leq v)$ とするとき, これら 2 つの点の間に表面が存在するのは

$$u \in (-\infty, t] \tag{2}$$

$$v \in [t, \infty) \tag{3}$$

$$0 < \Big((-u) - (-t)\Big) + (v - t) < \alpha w \tag{4}$$

が成立する場合のみとする. ただし α (≥ 1) は曲面の陰関数場への変換が誤差を含む場合にも表面を正しく発生させるために導入する係数である. 陰関数場が格子点サンプリングされた値を線形補間した値で与えられる場合, 空間内の点における関数値は最大格子点間隔の半分程度の誤差を含むと考えられるため, 実験では $\alpha = 1.5$ を使用した. 上の 3 つの式を満たすとき, 表面が存在する位置は

$$q = \frac{t - u}{v - u} \tag{5}$$

で求められる. ただし q $(\in [0, 1])$ は表面の位置が, $q = 0$ の時に値が u の格子点上に, $q = 1$ の時に値が v の格子点上になるように正規化されている.

次に表面に分岐がある場合に関して, 従来の陰関数表現では分岐部分において接続している表面の一部が除去されてしまう (図 1(a-2)). これは, 連続な実数値関数を用いて定義された陰関数ボリューム中では分岐を持つ表面の位置を正しく表現できないことが原因である.

曲面が多様体の場合, 曲面上の任意の点の近傍において面の表裏を定義することが出来る. したがって表面上の任意の点は, 最近傍点がその表側にある領域と裏側にある領域の 2 つの領域に挟まれる部分に存在し, このような領域分割を符号付距離関数を用いて表現することが出来る. ところが表面に分岐があり非多様体になっている場合には面の裏と表を定義することは出来ないため, 特に分岐線上の点の周辺領域を最近傍点の面の方向から 2 種類に分類することはできない.

入力の曲面を表裏の判別が出来る曲面パッチに分解しその表裏にそれぞれ番号を振ってこれらを区別することにより, 最近傍点の面の番号を用いて空間を複数領域に分類すると, 分岐のある曲面を陰関数表現することが可能になる (図 1(b-2)). 以下にその領域分割のアルゴリズムを示す.

1. 入力の非多様体曲面を分岐線上に沿って分割し, それぞれ分岐を持たない曲面パッチに分解する.

2. 得られたパッチに順に番号を振る. パッチの表裏を区別し, 表面の番号を i^+, 裏面の番号を i^- とする.

3. 空間を格子点 p でサンプリングし，格子点に曲面への Euclid 距離 $d_E(p)$ と最近傍点の面の番号 $i(p)$ を割り当てる．

4. 各格子点 p に対して, 6 隣接点 p_n で $i(p_n)$ を調べ, $i(p) \neq i(p_n)$ であるような $\Big(i(p), i(p_n)\Big)$ の組を列挙する．

5. 上で作られた番号の組を別の新たな番号で置換する．ただし置換の結果，最初に i^+ と i^- であった番号が同じ番号になる場合にはその組み合わせに対する置換を行わない. 最終的に番号が 0 から順に並ぶようにする．

6. 置換表に従い各格子点 p での領域番号 $i(p)$ を書き換える．

このようにして得られたボリュームの領域番号 $i(p)$ と, 各ボクセルにおける表面への Euclid 距離 $d_E(p)$ から, 実数値の陰関数ボリュームを構成する. そのために, 符号付距離を拡張した, 領域付距離を定義する. 領域付距離における「領域」とは符号付距離における正と負の 2 種類の「符号」を任意の数に拡張したものであり, 距離を表す実数空間を領域分割することにより実現する. 分割された領域の幅を D_s とする時，領域 i に含まれる距離 d_s^i は

$$d_s^i \in \Big[D_s i, D_s(i+1)\Big) \tag{6}$$

である．

以上のような領域付距離は以下の手順で計算できる. まず $D_s = 2^B$ と置き，$B = B_{max} - \lceil log_2 n \rceil$ から領域の大きさを決定する．ただし n は領域の数である．B_{max} は十分に大きい数であり, 計算機上で表現する場合には領域付距離を表す変数のビット数として与えることができる. このとき各ボクセルの位置 p で, $d_E(p)$ と $i(p)$ から領域付距離 $f_s(p)$ を次の式で計算する．

$$f_s(p) = \min(d_E, 2^B - \epsilon) + 2^B i(p) \tag{7}$$

ただし $\epsilon(> 0)$ は $f_s(p)$ が式 (6) の半開区間に含まれるように $d_E(p)$ を切り捨てるための微小な正の実数である．実験では ϵ を格子点間隔，$B_{max} = 8$ とした．

領域付距離関数は連続関数ではないため, その値を線形に補間することは出来ない．そこで，与えられた領域付距離場中の領域数を n, 距離場の 2 つのサンプル点の間隔を w, 値をそれぞれ u, v $(u \leq v)$ とするとき，これら 2 つの点の間に表面が存在するのは

$$u \in \Big[2^B i, 2^B(i+1)\Big), \tag{8}$$

$$v \in \Big[2^B j, 2^B(j+1)\Big), \tag{9}$$

$$0 < (u - 2^B i) + (v - 2^B j) < \alpha w, \tag{10}$$

が全て成立する i, j $(0 \leq i \leq j \leq n-1)$ が存在する場合のみとする. $\alpha \geq 1$ は式 (5) で用いた係数である. このとき表面が存在する位置は

$$q = \frac{u - 2^B i}{(u - 2^B i) + (v - 2^B j)} \tag{11}$$

で求められる. q の定義は式 (5) と同様である．

3　陰関数曲面の可視化

3.1　スライスを用いたボリュームレンダリング法

陰関数ボリュームに対して ray casting 法 [4] によるボリュームレンダリングの手法を適用することで陰関数曲面を直接, 描画することが可能である. ray casting 法では高画質な描画が可能である.

ray casting は任意の視線上での関数値のサンプリングが必要であるため比較的コストの高い計算であるが, Lacroute らはボリュームを x, y, z 軸と垂直な 3 組のスライスの集合で表現して (図 2 左), ボリュームのサンプリングをスライス平面上のみで行うことで計算効率を上げて描画を高速化する手法を提案した [3]. この方法はポリゴンとテクスチャマップを用いて効率的に実装することが可能である [1]. この手法では, ボリュームのスライスを透明度付き 2D テクスチャとしてポリゴンにマップし, 得られた半透明ポリゴンの集合を視線に対して奥の方から順に α ブレンドしながら表示する. この表示処理ではグラフィクスカードの描画加速を利用できるため, 高速に描画できる. またグラフィクスカードを使う場合には, 3D テクスチャ機能を利用することで, スライスの方向を固定せずに常に視線方向と垂直なポリゴンを発生させて描画することで (図 2 右), 多少の速度の減少と引き換えにより高画質な描画を行うことも可能である. 本報告における実験では, 視線と垂直なポリゴンを用いる方法を利用した.

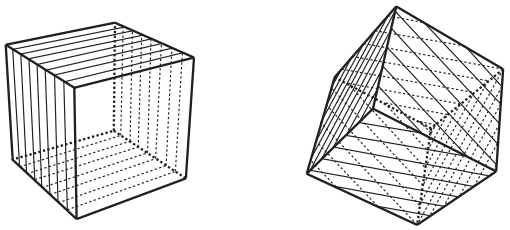


図 2: 左：ボリュームの軸に沿ったスライス生成. 任意の方向から表示する場合には x, y, z の 3 種類を持ち, 最も視線と垂直なものを選択する. 右：描画面と垂直なスライス生成. 視点変更のたびにスライスを再生成する.

3.2 Pre-integration 法 [2]

スライスを使ったボリュームレンダリングの大きな問題点は, ボリュームの値の補間が, スライス上では任意の解像度で行われるのに対し, スライスと垂直な方向では固定のスライス間隔でしか行われないため, 補間精度の不一致によりモアレと呼ばれる縞状の不具合や亀裂が観測される点である. この問題は, 発生させるスライスの枚数を, 必要な補間精度に応じて変更することにより回避可能であるが, 描画速度とスライス枚数は反比例の関係にあり, 多くの場合モアレが気にならない程度にスライスを発生させることは現実的ではない.

この問題に対して Engel らは, 各スライスに, そのスライス上でのボリュームの値だけでなく, 隣接するスライス上でのボリュームの値も同時にマップし, スライス間に存在するボリュームの影響も考慮することでモアレを除去する, pre-integration 法を提案した.

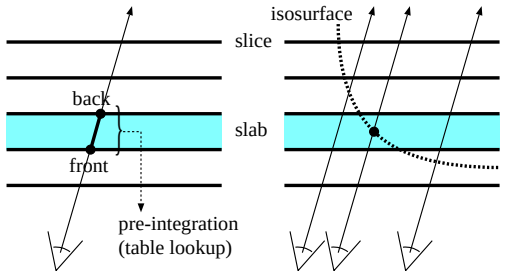


図 3: 隣接するスライス上のテクスチャを利用.

図 3 左のようにボリュームをスライス集合で表現し, front で示されるスライス S_{front} をポリゴンで描画する状況を考える. 従来のスライス法ではこのポリゴンに対し, スライス上でのボリュームの値を表すテクスチャ T_{front} をマップして表示を行うため, スライス間のボリュームの値は描画結果に影響しない. 一方, pre-integration 法では T_{front} と同時に back で示される隣接スライス S_{back} 上のテクスチャ T_{back} もマップし, 2 つのテクスチャを用いて S_{front} と S_{back} で囲まれる領域のボリュームの値を補間して表示する. その結果, スライスが存在しない領域も含めてボリューム全体の影響を描画

結果に反映させることが出来る. 実際にある視線での見え方を描画する時には, 視線と S_{front}, S_{back} の交点を計算して, front 側のテクセル値 t_{front} と back 側のテクセル値 t_{back} から，スライス間で視線上にあるボリュームがこの位置で観測される色と透明度に与える影響を計算して，ポリゴン上に表示する.

例えば図 3 右にあるように, スライスを用いたボリュームレンダリングの手法で，表面を表示する状況を考える. 図の中央の視線での見え方を考えるとき，従来のスライスを用いた手法ではスライスが表面上にはないため表面が描画されない．一方, pre-integration 法ではスライス間のボクセル値の変化を考慮することが出来るため, 少ない枚数のスライスを用いた描画でも正しい結果が得られる.

t_{front}, t_{back} の組から，観測される色と透明度を計算する処理を前もって行っておき, 結果を pre-integration table と呼ばれる早見表として保存しておくことで計算量を減らすことが可能である. またその表は 2 次元テクスチャとしてグラフィクスカード内に保存しておき，描画時にテクスチャ合成機能を利用して参照することにより高速に描画処理できる.

ボリュームはフォン・シェーディングによって照光処理される．そのため，ボリュームとして与えられるスカラー場の勾配をあらかじめ計算しておく必要がある．各ボクセルにおけるスカラー値 s と曲面の勾配 (u_x, u_y, u_z) が与えられるとき，ここからピクセル値 (s, u_y, u_z, u_x) を持つ 3D テクスチャを生成し，これを読み込む．描画時にはマルチテクスチャ機能を利用し，3D テクスチャをテクスチャユニット 0 と 1 に読み込み, ユニット 0 から t_{back} が，ユニット 1 から t_{front} が取り出せるようにテクスチャ座標を与える. pre-integration table は 2D テクスチャで表現して t_{front},t_{back} で与えられる任意のスカラー値の組に対する色を `RGB` チャネルに，透明度を `A` チャネルに割り当てる. ただし照光計算の際にスライス間で勾配を補間するために，`B` チャネルに勾配を補間する係数を割り当て，色は実際には `RG` の 2 チャネルのみで表現する．このテクスチャをユニット 3 に読み込むと，このユニットに対するテクスチャ座標がグラフィクスハードウェアによって (t_{front}, t_{back}) に設定され，pre-integration table が参照される.

3.3 陰関数曲面の可視化手法

領域付距離場ボリュームで表された陰関数曲面を pre-integration 法で描画するためには, 陰関数ボリュームを 3D テクスチャとして使える形に変換し, t_{front}, t_{back} からなる任意の領域付距離の組に対して, それらの間の描画色を決定するための pre-integration table を作ればよい.

まず陰関数ボリュームに関して, 領域付距離は一般に実数で与えられるが, 現行のグラフィクスカードでテクスチャとして使うために距離を 8bit 整数 (0–255) で量子化する．またシェーディングに必要な勾配は，陰関数場の生成の際に元の曲面の勾配から計算し，同様に量子化しておく. pre-integration table は 256×256 のサイズの 2 次元の `RGBA` テクスチャとして作成する. `RG` の 2 つのチャネルに割り当てる色は，領域付距離の距離の組ごとに任意の色を設定でき, 表面を均一に描画する時には単一色を与える. `A` チャネルには, スライス間に面が存在しない場合には `A`=0(透明), 存在する場合には `A`=1(不透明) とする. 最後に `B` チャネルに割り当てる補間係数として式 (11) から計算される, $q \in [0, 1]$ を 8bit 整数で量子化した値を用いる.

例として図 4 左図に示す非多様体曲面を陰関数表現して可視化することを考える. このモデルは球面が平面にはめ込まれた形の非多様体形状であり, 空間は 3 つの領域に分類できる. これに対応する pre-integration table は右図の通りである. 表の横軸が $t_{front} \in [0, 255]$, 縦軸が $t_{back} \in [0, 255]$ に対応し, 表の各点の色が描画される表面の色を色がない部分は表面が存在せずに透明色で描画される組み合わせに対応する．色の濃さは勾配の補間のための係数に対応している.

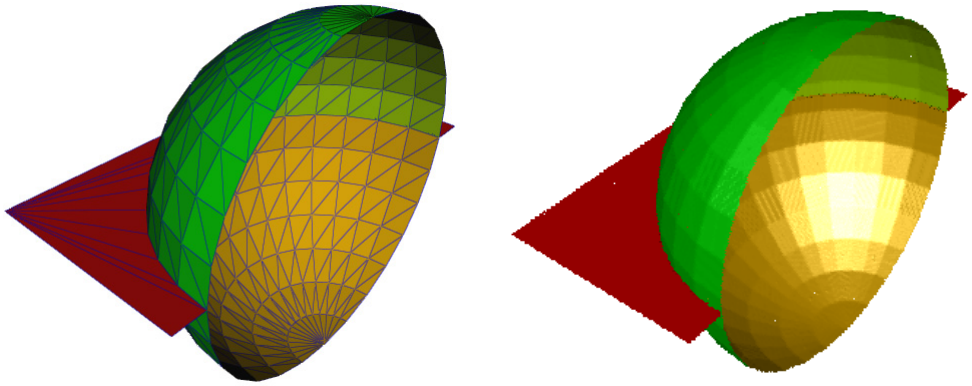


図 4: 描画結果. 上段：入力として与えた三角形メッシュモデル. 下段：陰関数表現に変換した後, 可視化した結果.

4　実験結果

可視化の実験に利用した計算機は Pentium4 1.7GHz, 主記憶 1.0GB, GeForce4 Ti4600 搭載グラフィクスカード, VRAM 128MB の PC である. 図 4 は非多様体表面モデルを陰関数表現に変換し, 可視化した結果である. 左が入力として与えたパラメトリック表現された表面モデルで，領域付距離場を用いて 256^3 サイズの陰関数ボリュームに変換し, 可視化した結果が右である. この表面は境界線と表面の分岐をもつ非多様体三角形メッシュであるが, 本手法では境界や分岐が正しく表現できることがわかる. 描画時の色は, 表面が存在する領域の対ごとに, 表面と裏面を区別して設定することが可能である. この例では全ての組み合わせに異なる色を設定している.

5　まとめと将来課題

本報では, 陰関数曲面を定義する距離関数と補間関数として不連続関数を用いることにより, 陰関数表現で非多様体曲面を扱えるように拡張した. またこのように定義された多様体・非多様体陰関数曲面を高速に描画する手法を提案した.

実験で用いた入力のパラメトリック曲面は三角形メッシュのみであるが, 提案手法は自由曲面にも適用可能である．ただし自由曲面を多様体パッチに分割するアルゴリズムなどの細かい変更は必要であると予想される. また形状の変形や集合演算を行う際には曲面の陰関数表現が有効であるが, 提案手法で定義された非多様体陰関数曲面表現に対してもこれらの処理が適用可能であるか考察する必要がある.

また本手法ではグラフィクスハードウェアの制限により，陰関数を 8bit 整数で量子化したが，これが原因で形状を正確に表示することができない．今後，32bit 浮動小数点テクスチャが利用可能なハードウェアが登場するため，これらを用いた実験を行う予定である.

従来の陰関数曲面では取り扱うことの出来る形状に制限が多く, 表面の境界と表面の分岐という

大きな問題に関して, 提案手法で解決することが可能である. しかしこれ以外にも多くの制約があり, 例えば, 向きつけ不可能な曲面の取り扱いなどにも拡張することが考えられるが, これらは今後の課題である.

参考文献

[1] High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading, High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading, High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. Two-phase perspective ray casting for interactive volume navigation, 1997.

[2] year, year. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading, pp. 9–16, 2001.

[3] year. Fast volume rendering using a shear-warp factorization of the viewing transformation, 1994.

[4] year. Direct 2D display of 3D objects. IEEE Mag. Computer Graphics and Applications, 1984.