陰関数表面表現の非多様体への拡張，及びその高速な可視化

山崎 俊太郎
東京大学，理化学研究所
shun@cvl.iis.u-tokyo.ac.jp

加瀬 究
理化学研究所
kiwamu@riken.go.jp

池内 克史
東京大学
ki@cvl.iis.u-tokyo.ac.jp

概 要

3 次元表面形状の表現形式の一つである陰関数表現は，広く用いられているパラメトリック表現に対して変形や集合演算などの位相的な処理の点で多くの利点を持つが，多様体形状しか扱うことが出来ないため適用範囲が狭い．本論文は，表面を定義する陰関数を不連続関数に拡張することで，非多様体形状を陰関数表現する手法を提案する．これにより表面モデルをパラメトリック表現と陰関数表現の間で相互に変換することが可能となり，多様体，非多様体にかかわらず，処理に適した表現形式を利用することが出来る．また，ボリュームレンダリングの手法を応用して陰関数表現された表面形状を効率的に描画できることを示し，表面形状の複雑さに依存することなく，陰関数表現された多様体，または非多様体の表面形状を高速に描画する方法を提案する．

キーワード: 陰関数表面, 非多様体, 領域付距離, 頂点生成表, pre-integration，ボリュームレンダリング

1 はじめに

計算機を用いて 3 次元物体を表現する際には，物体の表面の座標値を用いて形状を定義する，**パラメトリック表現**を使うのが一般的である．仮想現実感モデルや娯楽の分野でしばしば用いられるポリゴン表現，または CAD で用いられる自由曲面モデルはパラメトリック表現の一種であり，可視化や形状の最適化処理をするのに適している．

一方，表面の混合，変形，集合演算といった位相的な処理を行う場合には，もう一つの表現形式である**陰関数表現**を用いるほうが，処理が頑健かつ容易になることが多く，近年ではモデリング，シミュレーションなど多くの分野で使われるようになりつつある．陰関数表現されたデータは文字通り関数の形で保持することも可能であるが，複雑な処理を計算機内で効率的に処理するためには，関数をサンプルしてボリュームの形で保持しておくと都合が良い．これにより形状を表す関数場にデジタル信号処理の理論を適用することが可能となり，またパラメトリック表現への変換も容易になる．ボリュームは等間隔に並んだボクセルの 3 次元配列で定義されるため，ボリューム表現された陰関数表面は表面の形状に拠らず一定のコストでデータを処理でき，また解像度やサイズを変更した際の必要処理量がスケーラブルであるという利点がある．

表面形状のパラメトリック表現と陰関数表現にはそれぞれ利点，欠点があり，必要としている処理に応じて形式を選択できることが望ましい．

陰関数表現をパラメトリック表現に変換する手法は多くの研究者によって研究されており [9, 1]，特に三角形メッシュへの変換に関しては，任意の陰関数表面を，安定にかつ高速にポリゴン化する手法が提案されている．

一方，パラメトリック表現を陰関数表現に変換するためには，距離変換を用いて符号付距離場を生成する手法が有効である．しかしこの方法では必ずしも変換が成功するわけではなく，陰関数表面に変換することにより形状が大きく変更されてしまう場合がある．この問題の原因は二つあり，一つは陰関数表現に変換する際の再サンプリングによって信号が劣化するという点，もう一つは通常の定義の陰関数表面で扱える表面の種類は 2 多様体に限定されているという点である．前者の問題はサンプリング率を上げることによって回避可能であるが，後者の問題は根が深く，陰関数の表現そのものを拡張しないと解決できない．

表面形状の陰関数表現では，表面を 3 次元空間で定義される実数値関数の等値面として定義するため，空間内の各点は実数値が表面上での値より小さい領域と大きい領域の 2 つに分類され，その境界部分に表面が存在することになる．従って，元の表面上のある点で，その近傍がちょうど 2 つの領域には分類できない場合，その点や周辺の空間における関数値は未定義となり，矛盾なく陰関数場を構成することが出来ない．この問題が発生する例として分岐や境界を持つ表面形状が挙げられ，このような形状は非多様体と呼ばれる．

実数値関数による陰関数表現で非多様体を扱うことが出来ないという問題は，関数を多値化することによって解決することが出来ることが理論的に示されている [13, 10, 12] ．ある多値化された関数を用いて陰関数場が構築されているとき，そこからパラメトリック表現された非多様体表面を抽出する手法は研究されている．

しかしこれらの手法では非多様体の定義が曖昧であり，扱える表面形状の種類は不明である．また多値化された関数の定義は一意ではなく，与えられた表面形状から，特別な情報なしにそのような関数，また関数場を構成することは困難である．

本論文は，陰関数表現で扱うことの出来る非多様体の定義を明らかにした後，表面形状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．その際，入力として与えられるパラメトリック表現された表面形状から，領域付距離と呼ばれる距離関数を決定し，それを用いて領域付距離場と呼ばれる陰関数場を生成する方法を述べる．

表面形状の陰関数表現はそれ自体が表面を表すのに十分な情報を持っているが，表現形式が直感的でないために，与えられた表面の形状を想像する事が困難である．そこで陰関数表面を可視化する手法が重要になってくるが，従来の手法では非常に時間がかかり，また非多様体への拡張が困難であった．

そこで我々はボリュームレンダリングの手法を応用して陰関数表面を可視化する手法を提案する．描画にはグラフィクスカードの機能を利用することで対話的な速度で描画処理を行うことが可能であり，また処理速度は表面の形状に寄らず一定である．この手法では通常の多様体表面だけでなく非多様体表面も扱うことが出来る．

まとめると，本論文の主な貢献は次の 2 点である．

- 陰関数表面表現で扱う非多様体の定義を明確にし，表現に必要な関数の定義，また陰関数場を与えるボリュームの構成法を提案する．
- 上記の表現で表された多様体，非多様体の陰関数表面を，ボリュームレンダリングを応用することで高速に描画する手法を提案する．

以降の本論文の構成は以下の通りである．まず第 2 節で，陰関数表面の多様体への拡張，また陰関数表面の高速な可視化手法に関して関連研究を述べ，本手法の優位性を示す．第 3 節で陰関数表面の多様体への拡張，第 4 節で陰関数表面の高速な可視化手法に関して述べた後，第 5 節で実験結果を示し，最後に第 6 節でまとめと将来課題を述べる．

2 関連研究

2.1 非多様体形状の陰関数表現

表面形状の陰関数表現 [2] は形状モデリングの分野で広く研究されてきており，その応用はモデリング，変形，アニメーション，混合，描画など多岐にわたる．ただし従来の研究では実数値関数を用いて陰関数場を構成し，その等値面として表面を定義しているため，多様体形状しか扱うことが出来なかった．

Rossignac [13]，Muuss [10]，Paoluzzi [12] らによって指摘されているように，空間を複数領域に分類すれば非多様体を陰関数表面の枠組みで扱うことは可能である．この考察に基づき，Bloomenthal らは空間を四面体分割してポリゴン化することにより複数領域へ分類された陰関数場から非多様体の表面をポリゴン化する手法を提案した [3]．この手法は陰関数場が与えられたときに多様体ではない性質を持つ三角形メッシュを生成することが可能だが，この手法で扱うことの出来る非多様体の定義は曖昧である．また一般にパラメトリック表現で与えられる表面形状から，この手法が使える形式の陰関数場を構成する手順は自明ではない．そこで本論文では非多様体表面の定義を明確にし，パラメトリック表現された表面形状を陰関数表現に変換する手法を提案する．

2.2 陰関数表面の可視化

陰関数表面は光線追跡法 (ray tracing) による可視化と相性が良く，高精度，高画質に可視化することが可能である [2]．ただしこの手法は処理に時間がかかり，対話的な速度で描画する場合には解像度を大きく下げる必要がある．

陰関数表現をパラメトリック表現の一つであるポリゴンメッシュに変換することにより，既存のライブラリやハードウェアを使って高速に描画することが可能になる．Bloomenthal らは表面を表す陰関数場の空間を四面体分割し，各四面体ごとにポリゴン化する手法を提案している [1]．またこの手法は非多様体に対しても拡張できる [3]．一方 Lorensen らによって六面体分割を用いて陰関数場をポリゴン化する手法が提案されているが [9]，これも非多様体に拡張する研究がなされている [6, 17]．一度ポリゴンに変換する手法は，可視化処理の前に関数場からポリゴンを生成する必要があるため，変形などの処理によって関数場が変更される場合には，そのたびにポリゴン化を行う必要があり，対話的な可視化処理を行うことが困難である．

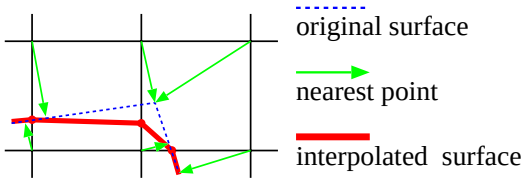


図 1: 空間中の各点で，表面への (符号付) 距離を計算しておき，それらの値が構成する場の等値面を抽出することでもとの表面を再構成することが可能である．このような表面表現を陰関数表現と呼ぶ．

そこで Witkin らは particles を用いて表面を可視化する手法を提案している [16]．しかしこの手法では，表面の形状が複雑になるほど必要な particle が増え，処理速度が減少する．また非多様体へ拡張された陰関数表面に対して適用できないという問題がある．

我々の提案手法では陰関数場をボリュームと考え，ボリュームレンダリングの手法を応用して陰関数表面を可視化する．形状の複雑度に無関係に描画することが出来るため，描画速度は一定である．またグラフィクスハードウェアのピクセルシェーディング処理を用いることにより大幅な高速化が可能であり，非多様体への拡張も可能である．

3　非多様体表面の陰関数表現

3.1　陰関数表面と符号付距離場

ある表面 $S \subset \mathbf{R}^3$ が与えられたとき，この表面 S の陰関数表現とは，ある実数値 t と実数値関数 $f(p)$ で

$$p \in S \quad \Leftrightarrow \quad f(p) = t, \tag{1}$$

を満たすような組 $\{t, f(p)\}$ のことである．ただし $p \in \mathbf{R}^3$ は 3 次元空間内で位置を表す点である．一般性を失うことなく $t = 0$ としてよく，通常 f のみを用いて表面を定義する．形状に対する処理を効率よく行うためには，f を関数の形ではなく，3 次元空間で等間隔にサンプルされたボリュームの形で用いると都合が良い．以降では，関数場はサンプルされたボリュームを考え，関数の値はボクセル値として保持されているものとする．

f は p と S の距離として定義できるが，この定義の方法は一通りではない．ボリューム化された関数場の場合，ボクセルの中心が表面上にある保証はないので，ボクセル中心の間の関数値は周辺の 8 つのボクセルの値から 3 次線形補間で計算する．従ってボクセル間にある表面位置を，補間によって決定できる距離であることが必要である．このような距離で最も広く使われているものとして，**符号付距離**がある．この関数は，ある点 p の S 上の最近傍点までの距離に，最近傍点が面の表側である場合には正の符号を，裏面にある場合には負の符号を付けた値を返す．2 次元における符号付距離場の例を図 1 に示す．

3.2　非多様体表面

3 次元空間で，表面 S が **(2-) 多様体**であるとは，表面上の任意の点における無限に小さい近傍が円盤と位相同相であることをいう．言い換えると，表面上の任意の微小領域に対してちょうど 2 つの座標軸をもつ局所座標を張ることができることを意味する．

逆に，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 3 つ以上取れる場合と考えることが出来る．即ち表面が非多様体的であるとは，次の 2 つの場合に分けられる．

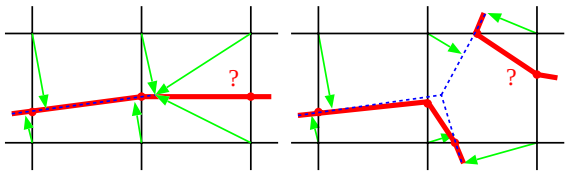
1. **表面が不連続である場合:**
   表面上の点 $p \in S$ で，少なくとも 1 つ，その近傍領域の局所座標系で独立な座標軸を 1 つ以下しか取れないものが存在する．これは，表面が境界を持つ場合にその境界上で発生する．例を図 2 の左図に示す．

2. **表面の接続が曖昧である場合:**
   表面上の点 $p \in S$ で，少なくとも 1 つ，その近傍領域の局所座標系で独立な座標軸を 3 つ以上とることが出来るものが存在する．これは，表面が分岐を持つ場合にその分岐線上で発生する．例を図 2 右図に示す．

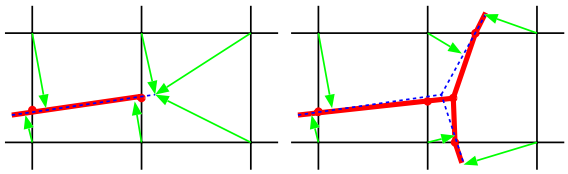
実際には，これらを満たす 3 次元空間内の点集合の種類には多くのバリエーションがある．そこで逆に，有限個の多様体を組み合わせた集合で，これらの 2 つの性質を持つものを非多様体表面と呼ぶことにする．したがって本論文で扱う非多様体表面は以下のように定義する．

多様体表面パッチ　3 次元空間内の点の集合 S_{mb} が多様体表面パッチであるとは，S_{mb} に埋め込まれた閉じた 1-多様体 B があり，$S_{mb} = S \cup B$，$S \cap B = \phi$，$S \neq \phi$ と分解でき，S_{mb} が $p \in S$ において 2-多様体であることを指す．B をこのパッチの境界と呼ぶ．

非多様体表面　S^i_{mb} を i 番目の多様体表面パッチ，B^i をその境界とするとき，非多様体表面 S_n とはある n



従来法で陰関数表現された形状



提案手法で陰関数表現された形状

図 2: 左上：元の表面に境界がある場合，符号付距離場で表された陰関数表面は元の表面から大きく変形してしまう．左下：面からの距離が一定以上の場合には面を再構成しないように変更することでこのような表面も表現可能である．右上：元の表面に分岐がある場合も符号付距離場では表現できない．右下：このような表面は領域付距離場を考えることで表現可能である．

に関して以下を満たすものである．

$$S_n = \bigcup_{i=1}^{n} S^i_{mb} \quad \text{s.t.} \quad \bigcup_{i=0}^{n} (S^i_{mb} - B^i) = \phi \qquad (2)$$

このような定義を満たす非多様体表面には，境界以外で交差しない自由曲面，三角形メッシュが含まれ，コンピュータグラフィクスや CAD で通常使われる形状表現としては十分な表現力を持つと考えられる．

3.3 領域付距離を用いた関数場ボリュームの生成

非多様体表面を陰関数表現で扱うときには，表面の不連続性と，表面の接続の曖昧性を解消すればよい．

表面の不連続性に関しては，表面からの距離が一定以上である点の付近には表面を再構成しないことで表現が可能である (図 2 左下)．面の接続の曖昧性に関しては，距離の定義を拡張する必要があるので以下その点を詳しく説明する．

多様体表面上では，任意の点において表面と裏面を定義することが出来る．したがって表面は，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する．符号付距離を用いることで空間を正の符号を持つ領域，負の符号を持つ領域の 2 種類に分類することが可能であるので，符号付距離場は多様体表面を表現することができる．ところが表面が非多様体の場合，1 つの点の周辺領域の分類は自明ではなく，2 つ以上に分類される場合があるため，この距離関数では非多様体表面を表現することが出来ない．

そこで複数の分類を可能にするために符号付距離を拡張した，**領域付距離** (segmented distance) を導入する．ここで「領域」とは符号付距離における符号 (正と負の 2 種類) を任意の数に拡張したものである．Bloomenthal [3] らが提案した分類手法ではこの領域情報と距離情報を分離して保持していたが，ここでは計算機を用いて処理することを前提に，別の方法を用いる．

計算機内部で距離を表現する場合，例えば一つの数を 32 個のビット列で表現することになる．32 ビット全体で表すことの出来る距離は 0 から $2^{32} - 1$ であるが，符号付距離ではその最上位の 1 ビットを符号を表現するのに用い，残りの 31 ビットで 0 から $2^{31} - 1$ の大きさの距離を表現していると考えることが出来る．そこで，複数の領域分類を可能にするために，最上位数ビットを領域を表すために使用することで領域付距離を表現することが可能である．

領域付距離関数は，必要な領域の数に応じて定義する必要があるため，表現する表面ごとに作り直す必要がある．そこで，入力として非多様体表面が与えられたときに，そこから領域付距離を定義し，それを用いて領域付距離場のボリュームを生成する方法を述べる．

1. 入力の非多様体表面を多様体パッチに分解する．ここで元の表面上で非多様体の特徴を持つ線分が全て境界となるように多様体パッチを生成する．ただし非多様体の特徴とは第 3.2 節で述べた 2 つのものである．

2. 得られた多様体パッチに順に番号を割り当てる．パッチの表裏を区別し，表面での番号を i^+，裏面での番号を i^- とする．

3. 3 次元空間をボクセル分割し，各ボクセルの中心位置で表面上の最近傍点を探索して距離を計算し，ボクセルにその距離と最近傍点の面の番号を領域番号として割り当てる．

4. 各ボクセルに対して，隣接ボクセルの領域番号を調べ，隣接している領域は同一領域であるとして 1 つの領域に統合する．ただし，任意の i に対して，i^+ と i^- の領域は統合化されないようにする．

5. 統合化が進まなくなった時点で，領域番号を 0 から順に振り直す．得られた領域の数を n とする．

6. 領域の定義に使用するビット位置 $B_s = B_{max} - \lceil log_2 n \rceil$ を決定し，各ボクセルに対し，ボクセルに割り当

てられている距離 d と新たに振られた領域番号 i から領域付距離 $d = d + 2^{B_s} i$ を計算し，そのボクセルにおけるボクセル値とする．ただし B_{max} はボクセル値を保持するために使える最大ビット数.

実験では入力として三角形メッシュを利用した．メッシュの頂点の座標に対し k-d 木を使った最近傍頂点の探索を行い，見つかった最近傍頂点に接続する三角形への距離を計算することにより距離 d を計算した．また $B_{max} = 8$ を使用した.

3.4 頂点生成表による補間処理

ボリューム化された陰関数場から表面の位置を再構成するとき，ボクセルの中心は必ずしも表面上にあるとは限らないので，ボクセル値の補間が必要になる．ボクセル値として符号付距離を使う場合には距離を線形補間することが可能であったが，領域付距離場を線形に補間することは出来ないので，補間のプロセスを制御する必要がある.

いま領域付距離場ボリュームが与えられた時，任意の 2 つのボクセル値 u, v に対してその間に表面が存在するのは，ボクセル幅を w，領域の数を n としたとき，$i, j = 1 \ldots n, i \neq j$ に対して

$$u \in [2^{B_s} i, 2^{B_s}(i+1)], \tag{3}$$

$$v \in [2^{B_s} j, 2^{B_s}(j+1)], \tag{4}$$

$$0 < (u - 2^{B_s} i) + (v - 2^{B_s} j) < w \tag{5}$$

が全て成立する場合であり，このとき表面が存在する位置は

$$p = \frac{u - 2^{B_s} i}{(u - 2^{B_s} i) + (v - 2^{B_s} j)} \tag{6}$$

で求められる.

この補間式を用いて任意の 2 つの距離に対してその間に面が存在するかどうかの判定を行うことにより正しい表面を再構成することが出来る．ただし，式が複雑であるため，元の表面形状が複雑になり，n が増えるに従ってこの判定式を処理するコストは指数的に増加する．そこで距離を離散化し，考えられる全ての距離の組に対して式 (6) の結果をあらかじめ計算しておき，早見表として持つことにより計算のコストを一定にする手法が高速化には有効である．この早見表を**頂点生成表** (vertex generation diagram) とよぶ．実験では距離を 8bit に離散化したため，$2^8 \times 2^8$ サイズの早見表を利用した.

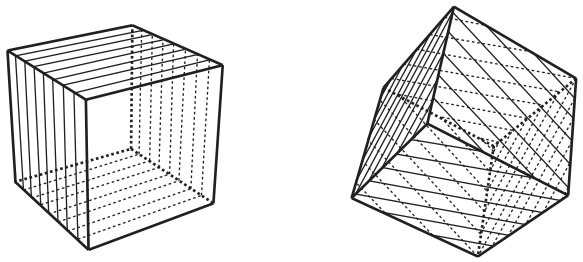


図 3: 左：ボリュームの軸に沿ったスライス生成．任意の方向から表示する場合には x-,y-,z-の 3 種類を持ち，最も視線と垂直なものを選択する．右：描画面と垂直なスライス生成．スライスは 1 組だけ持てばよいが，視点変更のたびにスライスを再生成する必要がある.

4 陰関数表面の可視化

4.1 Pre-integration 法

ボリューム化された陰関数場は，raycasting 法 [15] を用いてボリュームレンダリングすることで表示可能である．この手法は ray tracing の一種であり，ボリュームのサンプリングと積分計算のコストが非常に高く，必要のない計算を途中で打ち切る early ray termination 最適化 [8] や，octree[8] や k-d 木 [14] を使ってデータの局所的性質を利用する方法を使っても，対話的な速度では描画できない.

一方，Lacroute らによってボリュームを x-,y-,z-軸と垂直なスライスの集合で表現して (図 3 左)，ボリュームのサンプリングをこれらスライス平面上のみで行うように簡略化することでデータアクセス効率を上げて描画を高速化する手法が提案された [7]．この方法は，テクスチャマップされた半透明ポリゴン表示を使って描画することで，グラフィクスカードの描画加速を利用することができる [4]．また近年では 3D テクスチャ機能を使うことにより，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 3 右)，さらに高画質な描画を行うことも可能になった．これらの手法を使うと 256^3 程度のボリュームデータであれば通常の PC で対話的な速度でボリュームレンダリングを行うことが可能である.

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，ボクセル値の補間が，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，または亀裂として観測されるという点がある．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

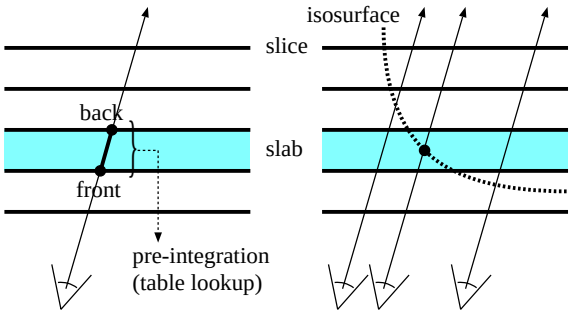


図 4: 左：現在のスライス (front) と隣接スライス (back) のボクセル値を利用して，スライス間のボクセル値を補間して表示する．右：形状表面がボリューム上にない場合にも面を表示できる．

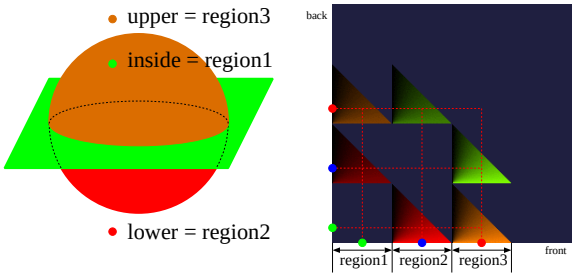


図 5: 左：非多様体モデルに対する領域分け．右：対応する頂点生成表．

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．

これに対し Engel らは，各スライスにそのスライス上のボクセル値に対応するテクスチャをマップするだけでなく，隣接するスライス上のボクセル値に対応するテクスチャも同時にマップし，それらのスライス間方向へのサンプリングも考慮することでモアレを除去する，pre-integration 法 [5] を提案した．図 4 左の状況を考えると，図中の front で示されるスライスに対して，そのスライス上でのボクセル値を表すテクスチャ T_{front} と，back で示される隣接スライス上のボクセル値を表すテクスチャ T_{back} を同時にマップする．これら 2 つのスライス上のボクセル値からスライス間のボクセル値を補間して描画することができる．任意の視線方向での見え方を描画するには，視線と front，back スライスの交点を計算し，front 側のテクスチャの交点におけるテクセル値 t_{front} と back 側のテクスチャの交点におけるテクセル値 t_{back} を用いて，スライス間の視線が通過する領域のボクセル値を補間し，それら全ての影響を数値的な積分から算出することにより，より高精度な描画を行う．

例として，ある表面モデルから生成した符号付距離場ボリュームを描画する状況を考える．いま元の表面が図 4 右のようにあったとすると，右端の視線，左端の視線において，t_{front} と t_{back} は同一の符号であるので，現在描画中のスライスには何も表示しない．しかし中央の視線では，t_{front} と t_{back} が異なる符合であることから，このスライス間には表面が存在することがわかるので，表面に対応する色を描画する．

従来のスライスを用いたボリュームレンダリング手法では図 4 右の中央の視線において，各スライスでスライス上のボクセル値のみから描画を行うため，スライスがちょうど表面上には乗っていない結果，表面を描画することが出来ない．一方 pre-integration を行うことでスライス間のボクセル値の変化を考慮することが出来，不十分な枚数のスライスを用いた描画でも正しい結果を得ることが出来る．

pre-integration を使ったボリュームレンダリングで必要な計算は，スライス同士の合成とスライス毎の計算の 2 種類に分けられるが，前者は一般的なグラフィクスライブラリ，ハードウェアで提供されるアルファ混合機能を使うことで行うことが出来る．後者に関しては t_{front} と t_{back} からその視線上の見え方を計算する pre-integration と呼ばれる計算が必要であるが，この計算をあらかじめ行っておき，結果を pre-integration table と呼ばれる早見表として持つことで描画を高速化することが出来る．またこの pre-integration table をテクスチャとして保持し，nVidia 社製の GPU GeForce3 のテクスチャ合成機能を使うことで大幅に高速化することが可能である．実装は [5] に詳細に記述されているためここでは省略する．

4.2 頂点生成表の利用

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，任意の領域付距離 t_{front} と t_{back} に対して，それらの間に表面が存在するか判定すればよい．したがってこの領域付距離に対応して生成した頂点生成表を pre-integration table として利用することで，表面を表示することが可能である．

非多様体の陰関数表面を pre-integration 法で可視化する際には，テクスチャとして与えられるボクセル値の領域付距離が線形補間されないように，不要な補間を抑制する必要がある．2 つの異なる領域付距離を線形に補間すると，結果として全く異なる領域に属する距離

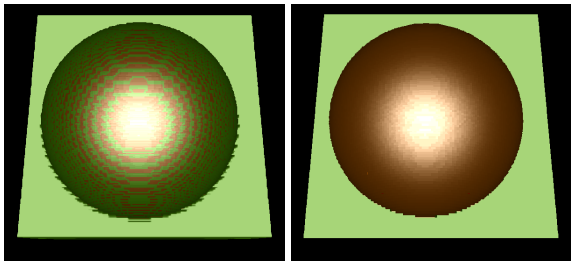


図 6: 左：領域付距離場の値を線形補間した場合の不具合．右：最近傍値をとることで不具合を防げる．

が得られたり，未定義領域の数値を返すことになる．

図 5 左に示す非多様体表面の可視化を考える．このモデルは球が平面に埋め込まれた非多様体形状で，3 つの領域に分類されている．対応する頂点生成表は図 5 右の通り．このとき，ボクセル値の線形補間を行って可視化した結果が図 6 左図である．線形補間によって本来存在しない領域に属する距離が生成され，本来存在しない表面がモアレ状に現れている．この不具合は，ボクセル値の線形補間を行わないことで解決することができ，結果として図 6 右の結果を得ることが出来る．

5　実験結果

我々は提案手法を PC 上で実装し，複数のモデルに対して実験を行った．可視化の実験に利用した計算機は CPU Pentium4 1.7GHz，主記憶 1.0GB，VPU GeForce4 Ti4600，VRAM 128MB の PC である．可視化のソフトウェアとして描画面に水平なスライスを用いて pre-integration を行うボリュームレンダラを作成し，pre-integration table として頂点生成表を組み込んだ．実装は Windows 2000 上の Visual C++ 6.0 を用いて行い，OpenGL グラフィクスライブラリと nVidia の OpenGL 拡張 [11] である Texture shader(`GL_NV_texture_shader`) 拡張，及び Register Combiners (`GL_NV_register_combiners`) 拡張を利用した．またこれらの設定は [5] を参考にした．

図 7 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．左図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，実際には三角形メッシュを使用した．これらのモデルを領域付距離場を用いて陰関数表面に変換し，64^3 サイズのボリュームを生成した．それを可視化した結果が中央図であり，右図が可視化の際に利用した頂点生成表である．

図 7 上段が，実物体の計測して得られた猫のモデルに本手法を適用した結果である．回転台を用いて計測したため，モデルの底面の形状は計測できておらず，したがって穴が開いている．そのため底面付近に表面の境界が存在し，非多様体表面になっている．このモデルを 2 つの領域を持つ距離場に変換し，可視化した結果が中央上段である．従来の陰関数表現ではこのような境界を表現することが出来なかったが，ここではうまく再現されている．

図 7 下段は，平面に球を埋めこみ，半分に切ったモデルに適用した結果である．モデルは境界を線を持ち，中央の球と平面部分との交差部分に面の分岐があるため，非多様体である．このモデルを 3 つの領域を持つ距離場に変換し，可視化した結果が中央下段である．境界や分岐が正しく再現されていることがわかる．

また 128^3 と 256^3 サイズのボリュームデータに対し可視化処理を行った際の描画速度を表 1 に示す．発生させるスライス枚数を変更することによって速度は変化するが，多くの実験結果で対話的な速度に必要とされる 30fps を達成できた．

6　まとめと将来課題

関数の不連続関数を考慮することで拡張し，陰関数表現を用いて非多様体モデルを扱えるように拡張した．またこのように定義された陰関数表面はボリュームレンダリングの手法を応用することで高速に描画可能であることを示した．

本論文では主に三角形メッシュを陰関数表現に直す実験を行ったが，本手法は自由曲面などにも適用可能であり，実証していく必要がある．また多くの研究者によって，形状の変形や集合演算を行う際の陰関数表面の有効性が示されてきた．非多様体形状にも適用可能な処理の種類を考察し，実際に陰関数場を変化させることで形状の操作が可能であるか調べるのは今後の課題である．

参考文献

[1] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, Vol. 5, pp. 341–355, 1988.

[2] J. Bloomenthal. *Introduction to Implicit Surface*. Morgan Kaufmann Publishers, Inc., 1997.

[3] J. Bloomenthal and K. Ferguson. Polygonization of non-manifold implicit surfaces. *Computer Graphics*, Vol. 29, pp. 309–316, 1995.

[4] M. Brady, K. Jung, H.T. Nguyen, and T Nguyen. Two-phase perspective ray casting for interactive volume navigation. In *Visualization 97*, 1997.

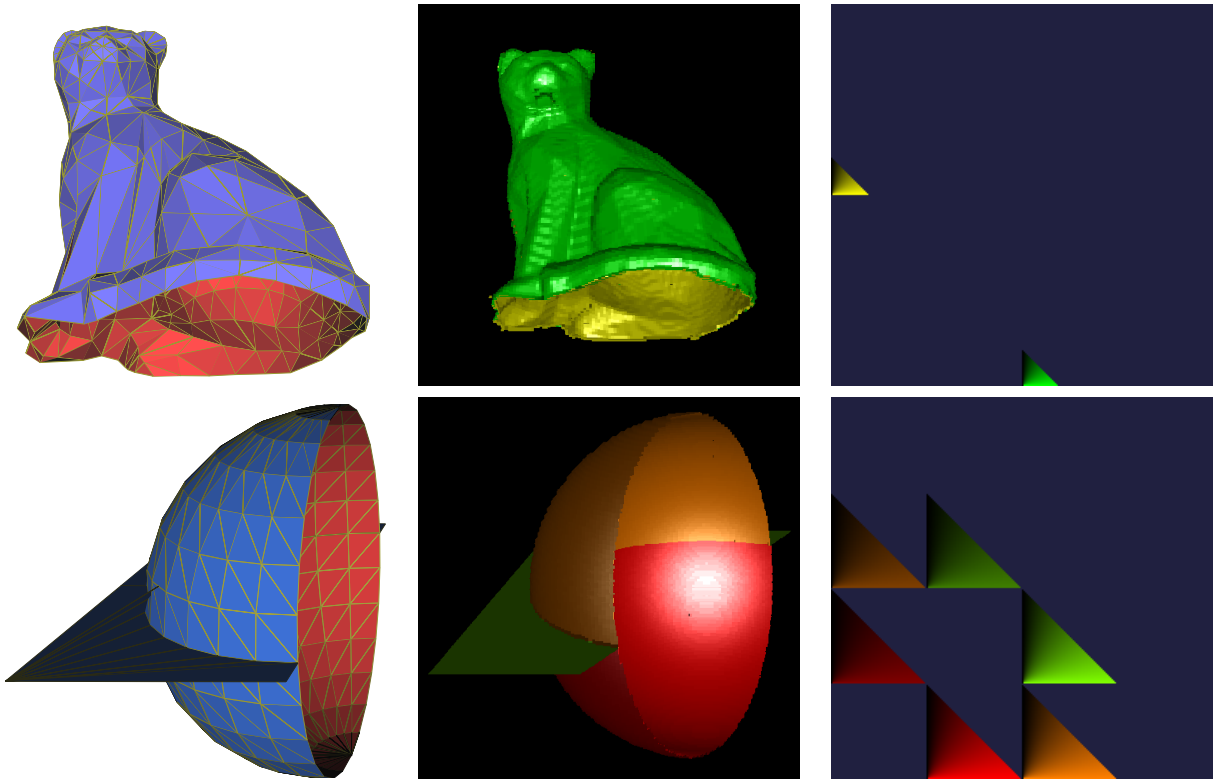


図 7: 描画結果．左列：入力として与えた三角形メッシュモデル．中央列：陰関数表現に変換した後，可視化した結果．右列：可視化の際に利用した頂点生成表．上段:境界のある非多様体表面モデル．下段：境界と分岐のある非多様体表面モデル．陰関数場として使用したボリュームのサイズはどちらも 64^3 である．

表 1: 描画速度

Size of volume	128^3				256^3			
Number of slices	32	64	128	256	32	64	128	256
FPS	>30	>30	>30	15	>30	>30	15	8

[5] K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Eurographics / SIGGRAPH Workshop on Graphics Hardware '01*, pp. 9–16, 2001.

[6] H. C. Hege, M. Seebas, D. Stalling, and M. Zockler. A generalized marching cubes algorithm based on non-binary classifications. Technical report, Konrad-Zuse-Zentrum fur Informationstechnik (ZIB), 1997.

[7] Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *SIGGRAPH '94*, 1994.

[8] Marc Levoy. Efficient ray tracing of volume data. *ACM Trans. on Graphics*, Vol. 9, No. 3, 1990.

[9] W.E. Lorensen and H.E. Cline. Marching cubes: a high resolution 3d surface reconstruction algorithm. In *SIGGRAPH '87*, 1987.

[10] M. Muuss and L. Butler. *Computer Graphics Techniques: Theory and Practice*, chapter Combinatorial Solid Geometry, B-Repsm and n-Manifold Geometry. Springer Verlag, 1990.

[11] nVidia corporation. *nVidia OpenGL specifications*.

[12] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transaction on Graphics*, Vol. 12, , 1993.

[13] J. Rossignac and M. O'Connor. SCG: a Dimension-Independent Model for Pointers with Internal Structures and Imcomplete Boundaries. In *Geometric Modeling for Product Engineering*, 1990.

[14] K. R. Subramanian and Donald S. Fussell. Applying space subdivision techniques to volume rendering. In *Visualization 90*, 1990.

[15] H. Tuy and L. Tuy. Direct 2d display of 3d objects. *IEEE Mag. Computer Graphics and Applications*, 1984.

[16] A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. In *SIGGRAPH '94*, 1994.

[17] Shuntaro Yamazaki, Kiwamu Kase, and Katsushi Ikeuchi. Non-manifold implicit surfaces based on discontinuous implicitization and polygonization. In *Geometric Modeling and Processing*. IEEE, July 2002.