

パラメトリックパッチとボリュームを用いた 陰関数表現による幾何ソリッド

三 浦 憲 二 郎†

この論文では、 n 次元の自由曲面で囲まれたソリッドの陰関数表現へ n 変数のパラメトリック関数を用いる定義法とその処理のための計算手法について論じる。パラメトリック関数は通常その定義域を限定して用いられ、定義域の外では関数値が急激に増大あるいは減少し、幾何モデルを定義するには適していない。そこで、本論文では関数クリッピング操作を新たに導入し、R関数に基づく集合演算に用いられる陰関数表現された立体プリミティブと同じように、パラメトリック関数により表現されたプリミティブを取り扱うことを可能とした。また、定義域が変更された場合に高速に逆マッピング（与えられた座標値に対応するパラメータ値の算出）を行う拡張 Bézier クリッピング操作を提案した。提案する手法の有効性を示すために、集合演算やオフセット、ハイパテクスチャ、モーフィングの例を示した。また、応用例として、トリミングされたパッチの三角形分解を示した。

Implicit Representation of Geometric Solids Using Parametric Patches and Volumes

KENJIRO T. MIURA†

We take an approach to specify and control implicitly defined n -dimensional free-form solid primitives using parametric functions of n variables. Outside of the domain, the parametric function value increases or decreases rapidly that is not suitable to design geometric objects. In this paper, we introduce a functional clipping operation which allows a free-form primitive to be treated as traditional implicit primitives in set-theoretic modeling based on R-functions. We also introduce an extended Bézier clipping operation for fast inverse mapping in the case of the changed domain. Examples of set-theoretic, offsetting, hypertexturing and metamorphosis operations on free-form primitives are given. Tessellation of trimmed parametric patches is explained as an application of our new geometric modeling method.

1. はじめに

形状モデリングやコンピュータグラフィックスの分野で、曲線や曲面を表現する典型的な方法はパラメトリック関数を用いた表現法である。パラメトリック表現の長所の1つは、制御点の位置を修正することによって曲線・曲面を会話的に設計・修正できることにある。曲線・曲面のもう1つの方法は陰関数表現であり、関数値が0となる点の集合によってそれらを表す。陰関数表現はオフセットやブレンディングといった重要な幾何演算に対して閉じているといった好ましい性質を持ち、CADやCG、アニメーションなどに広く利用されている。しかしながら、陰関数表現の欠点として、パラメトリック表現された立体よりも変形が容

易ではないことがあげられる。また、陰関数表現に用いられる立体プリミティブは通常、二次曲面やトーラス、超二次曲面（superquadric）、スケルトンに基づくモデルに限定されている。

これらの2つの表現をうまく組み合わせて使うというのがこの論文のねらいである。この論文では、 n 次元の自由曲面で囲まれたソリッドの定義に n 変数のパラメトリック関数を用いる。この方法は、区分（piecewise）代数曲面を生成したり^{2),5),8)}、厚い自由曲面シェルを定義する¹⁾のに利用されている。それらに対して、本論文では曲面パッチやシェルではなく、ソリッドプリミティブをモデリングし、それらをCSGモデラで使用する。その際に問題になるのは次の2点である。

- (1) パラメトリック関数の各々のパラメータ t の定義域は通常限定されている（たとえば、 $0 \leq t \leq 1$ ）。定義域の外では関数値は急激に増加したり

† 会津大学コンピュータ理工学部

School of Computer Science and Engineering, University of Aizu

減少したりする。定義域の外ではつねに負の値になるとは限らず、孤立した目的外のソリッドを生成してしまう可能性がある。そこで、本研究では関数クリッピング操作を導入し、定義域の外での関数の値を制御する。

- (2) 制御点の移動によるパッチやボリウムの定義域を変更することによって、より自由なモデリングが可能となる。しかしながら、この変更にとともに制御点の規則的な配置がくずれてしまう。制御点を関数が定義された平面、あるいは空間内で動かした場合、与えられた点での関数値を評価するためにパラメータを算出する逆マッピングを行わなければならない。本研究ではこの問題を解決するために、Bézier クリッピングを改良した拡張 Bézier クリッピングを提案する。

R 関数^{6),9)}を利用すれば、陰関数表現の集合演算を閉じることができる。これは集合演算を繰り返して得られるような複雑なソリッドが1つの関数で表現できることを意味する。そのような関数を適切に評価し、形状を精度良く表示するには多大な時間を必要とするが、この方法を用いるとブレンディングやオフセット、ハイパテクスチャ、衝突チェックといった問題に対して汎用的な解法が得られる⁴⁾。関数クリッピングと拡張 Bézier クリッピング操作の導入により、R 関数で集合演算されるソリッドプリミティブの1つとしてパラメトリック関数で表現された自由曲面プリミティブを取り扱うことが可能となる。提案する手法の有効性を示すために、集合演算やオフセット、ハイパテクスチャ、モーフィングの例を示す。また、応用例として、トリミングされたパッチの三角形分解を示す。

2. パラメトリックパッチとボリウムによる陰関数表現

2.1 ソリッドの陰関数表現

次式を満たす n 次元ユークリッド空間 E^n の閉じた部分空間を考える。

$$f(x_1, x_2, \dots, x_n) \geq 0 \quad (1)$$

ここで、 f は E^n 上で定義された連続した実数値関数とする。 f を定義関数と呼ぶ。不等式 (1) をソリッドの陰関数表現と呼ぶ。関数は様々な方法で定義することができる。たとえば、評価アルゴリズムとともに解析的に定義することもできるし、テーブル化された関数値とそれらの内挿法によって定義することもできる。少なくとも C^0 連続であることが関数に要求される性質である。不等式 (1) で表される n 次元の立体

は以下のように定義する。

$$f(\mathbf{X}) > 0 \quad - \text{立体の内部の点} \quad (2)$$

$$f(\mathbf{X}) = 0 \quad - \text{立体の境界上の点} \quad (3)$$

$$f(\mathbf{X}) < 0 \quad - \text{立体の外部の点} \quad (4)$$

ここで $\mathbf{X} = (x_1, x_2, \dots, x_n)$ は E^n の点である。

2つのソリッドの集合演算である「和」と「積」、ソリッドの内部と外部を「反転」する (negate) ために、Rvachev⁶⁾によって研究されたいわゆる R 関数は、以下のように定義される。

$$\begin{aligned} f_1 \cup_{\alpha} f_2 \\ = \frac{1}{1+\alpha} \left(f_1 + f_2 + \sqrt{f_1^2 + f_2^2 - 2\alpha f_1 f_2} \right) \end{aligned} \quad (5)$$

$$\begin{aligned} f_1 \cap_{\alpha} f_2 \\ = \frac{1}{1+\alpha} \left(f_1 + f_2 - \sqrt{f_1^2 + f_2^2 - 2\alpha f_1 f_2} \right) \end{aligned} \quad (6)$$

$$\neg f_1 = -f_1, \quad (7)$$

ここで $\alpha = \alpha(f_1, f_2)$ であり、次の条件を満たす。

$$-1 < \alpha(f_1, f_2) \leq 1,$$

$$\begin{aligned} \alpha(f_1, f_2) &= \alpha(f_2, f_1) = \alpha(-f_1, f_2) \\ &= \alpha(f_1, -f_2) \end{aligned} \quad (8)$$

$\alpha = 1$ の場合には、式 (5), (6) は次式となる。

$$\begin{aligned} f_1 \cup_{\alpha} f_2 &= \frac{1}{2} (f_1 + f_2 + |f_1 - f_2|) \\ &= \max(f_1, f_2) \end{aligned} \quad (9)$$

$$\begin{aligned} f_1 \cap_{\alpha} f_2 &= \frac{1}{2} (f_1 + f_2 - |f_1 - f_2|) \\ &= \min(f_1, f_2) \end{aligned} \quad (10)$$

これらの式は簡潔ではあるが、 $f_1 = f_2$ の点で C^1 連続性が保たれない。実用上最も有効なのは $\alpha = 0$ の場合であり、式 (5), (6) は、

$$f_1 \cup_{\alpha} f_2 = f_1 + f_2 + \sqrt{f_1^2 + f_2^2} \quad (11)$$

$$f_1 \cap_{\alpha} f_2 = f_1 + f_2 - \sqrt{f_1^2 + f_2^2} \quad (12)$$

となる。この場合には C^1 不連続となるのは2つの関数値がともに0となる点だけである。

形状モデリングにおいて「差」も重要な集合演算であるが、 f_1 と f_2 の「差」は「和」と「反転」を用いて、

$$\neg((\neg f_1) \cup_{\alpha} f_2) \quad (13)$$

と表せる。これに式 (5), (7) を適用すると、

$$\begin{aligned} \neg((\neg f_1) \cup_{\alpha} f_2) \\ = \frac{1}{1+\alpha} (f_1 - f_2 - \sqrt{f_1^2 + f_2^2 + 2\alpha f_1 f_2}) \end{aligned} \quad (14)$$

となる。同様に「積」は「和」と「反転」を用いて、

$$\neg((\neg f_1) \cup_{\alpha} (\neg f_2)) \quad (15)$$

と表される. 式 (6) は, この式に式 (5), (7) を適用しても得られる.

2.2 パラメトリックパッチとボリウム

様々な自由曲面パッチがこれまでに提案され, 複雑な自由曲面を持つ立体のモデリングに利用されている. パッチ上の 1 点は,

$$(x, y, z) = \mathbf{S}(u, v) \quad (16)$$

で与えられる. Bézier パッチや NURBS (non-uniform rational B-spline) 曲面は幾何計算に好ましい性質を持つのでよく利用されている. この論文では簡潔さを保つために多項式 Bézier パッチとボリウムを例として取り上げる. しかしながら, 他のパラメトリック関数, たとえば有理 Bézier パッチや NURBS 曲面, Gregory 型パッチを陰関数表現の定義に用いても理論上問題はない.

Bézier パッチ $\mathbf{S}(u, v)$ は以下のように定義される.

$$\mathbf{S}(u, v) = \sum_{i=0}^m \sum_{j=0}^n B_i^m(u) B_j^n(v) \mathbf{P}_{ij} \quad (17)$$

ここで $\mathbf{P}_{ij}$ は制御点と呼ばれる 3 次元の点である. B_i^n ($i = 0, \dots, n$) は Bernstein 基底関数であり,

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i \quad (18)$$

である. 同様に Bézier ボリウムは以下のように定義される.

$$\mathbf{S}(u, v, w) = \sum_{i=0}^l \sum_{j=0}^m \sum_{k=0}^n B_i^l(u) B_j^m(v) B_k^n(w) \mathbf{P}_{ijk} \quad (19)$$

ここで $\mathbf{P}_{ijk}$ は 4 次元の点である. この方法を n 次元に拡張するのは容易であり, $n+1$ 次元の点を用いればよい. 制御点の位置を変更することによって立体の形状を変更することができる.

2.3 ソリッドの定義

まず, 簡単な例として xy 平面上の単位正方形内に 2 次元ソリッドを定義する. もちろん陰関数表現に用いられる関数 $f(x, y)$ は 2 次元平面全体で定義されなければならない. そこで, 単位正方形で与えられる領域の外では関数 $f(x, y)$ の値がつねに負となるように, 新たに関数クリッピング操作を導入する. その後, より一般的な領域において定義できるように拡張する. 関数クリッピング操作については 3 章で説明し, この節では単位正方形で与えられる領域でのソリッドの定義について論じる.

定義域 $\{(x, y) | 0 \leq x, y \leq 1\}$ において陰関数表現 $f(x, y) \geq 0$ によって 2 次元ソリッドを定義するため

に, $\mathbf{S}(u, v)$ の x, y 座標値 $S^x(u, v)$ と $S^y(u, v)$ は,

$$S^x(u, v) = \sum_{i=0}^m \sum_{j=0}^n B_i^m(u) B_j^n(v) P_{ij}^x \quad (20)$$

$$= u, \quad (21)$$

$$S^y(u, v) = \sum_{i=0}^m \sum_{j=0}^n B_i^m(u) B_j^n(v) P_{ij}^y \quad (22)$$

$$= v \quad (23)$$

で与えられるように制御点を配置する. したがって,

$$P_{ij}^x = \frac{i}{m}, \quad P_{ij}^y = \frac{j}{n}. \quad (24)$$

$\mathbf{S}(u, v)$ の z 座標値, $S^z(u, v)$ を用いて 2 次元ソリッドを次式で定義する.

$$f(x, y) = S^z(x, y) = S^z(u, v). \quad (25)$$

定義域 $\{(x, y, z) | 0 \leq x, y, z \leq 1\}$ において陰関数表現 $f(x, y, z) \geq 0$ によって 3 次元ソリッドを定義するために, $\mathbf{S}(u, v)$ の x, y, z 座標値 $S^x(u, v)$ と $S^y(u, v)$, $S^z(u, v, w)$ は,

$$S^x(u, v, w) = \sum_{i=0}^l \sum_{j=0}^m \sum_{k=0}^n B_i^l(u) B_j^m(v) B_k^n(w) P_{ijk}^x \quad (26)$$

$$= u, \quad (27)$$

$$S^y(u, v, w) = \sum_{i=0}^l \sum_{j=0}^m \sum_{k=0}^n B_i^l(u) B_j^m(v) B_k^n(w) P_{ijk}^y \quad (28)$$

$$= v, \quad (29)$$

$$S^z(u, v, w) = \sum_{i=0}^l \sum_{j=0}^m \sum_{k=0}^n B_i^l(u) B_j^m(v) B_k^n(w) P_{ijk}^z \quad (30)$$

$$= w, \quad (31)$$

すなわち,

$$P_{ijk}^x = \frac{i}{l}, \quad P_{ijk}^y = \frac{j}{m}, \quad P_{ijk}^z = \frac{k}{n} \quad (32)$$

とする. $\mathbf{S}(u, v, w)$ の 4 番目の座標値である $S^w(u, v, w)$ を用いて 3 次元ソリッドを以下のように定義する.

$$f(x, y, z) = S^w(x, y, z) = S^w(u, v, w). \quad (33)$$

図 1 (a) は次数 7×7 の Bézier パッチを示し, 図 1 (b) は単位正方形内のパッチによって定義された 2 次元ソリッドの境界線 ($f = 0$ である輪郭線) を示す. 図 1 (c), (d) と図 1 (e), (f) は 2 つに分離したソリッドと内部に穴の空いたソリッドを示す. これらの図から 1 つのパッチを利用しながら様々なソリッドを表現できることが分かる.

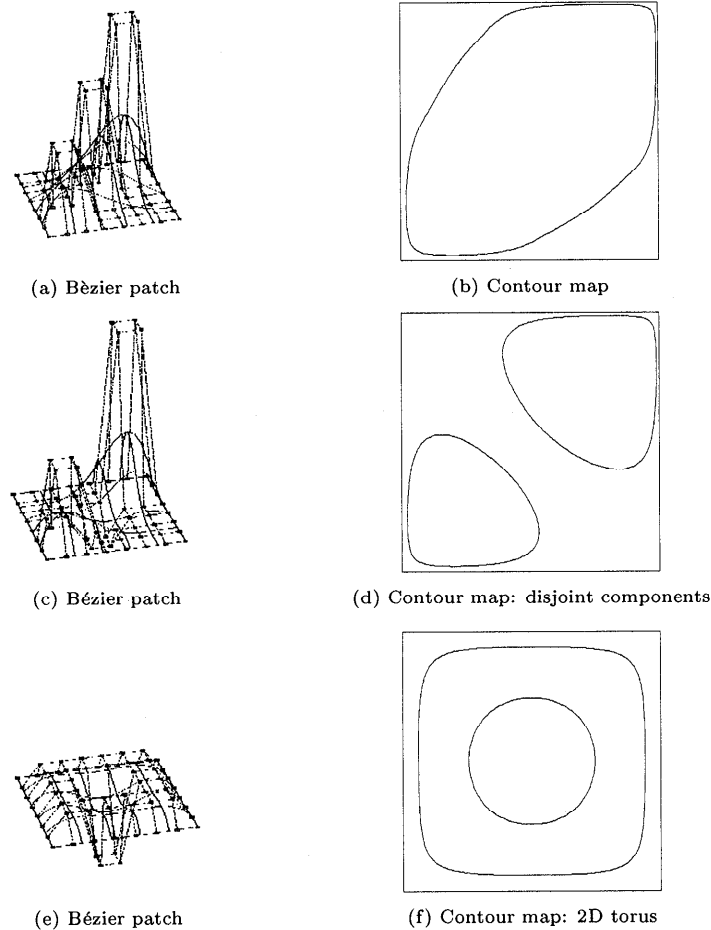


図1 2次元ソリッド例
Fig. 1 2D solid examples.

3. パッチとボリュームのクリッピング

3.1 関数クリッピング

図2(a)と(b)は単位正方形よりも大きな領域で定義されたBèzierパッチによる曲面とその輪郭線を示す。パッチは単位正方形内で1つの閉じた曲線を定義するように生成されている。これらの図から分かるように、単位正方形の外では関数値が急激に増加する領域があり、それらの領域において関数値を負にする必要がある。

1つの方法として単位正方形(3次元の場合には単位立方体)を境界箱として使用し、すべての応用に対してチェックすることが考えられる。しかしながら、この方法は定義域が変形された場合適用することができない。そこで、これまでの陰関数の取扱いとまったく同様にパラメトリック関数による陰関数を取り扱う

方法を考える。陰関数表現の利点の1つである均一性(uniformity)を保つために、次のような関数クリッピングを新たに導入する。

$S(u, v)$ をパラメトリックパッチを定義する関数とし、 $S(u, v) = S^z(u, v)$ とする。単位正方形とこの関数によって定義される陰関数表現された立体との「積」を考える。

$$S_{clip}(u, v) = S(u, v) \cap F_s(u, v) \quad (34)$$

ここで $F_s(u, v) = F_b(u) \cap F_b(v)$ は単位正方形を定義する関数であり、 F_b は次式で与えられ、各パラメータの単位セグメントを定義する。

$$F_b(t) = (1-t)t \quad (35)$$

ここで $\cap$ は「積」を意味し、たとえば式(12)を用いて、

$$f_1 \cap f_2 = f_1 + f_2 - \sqrt{f_1^2 + f_2^2} \quad (36)$$

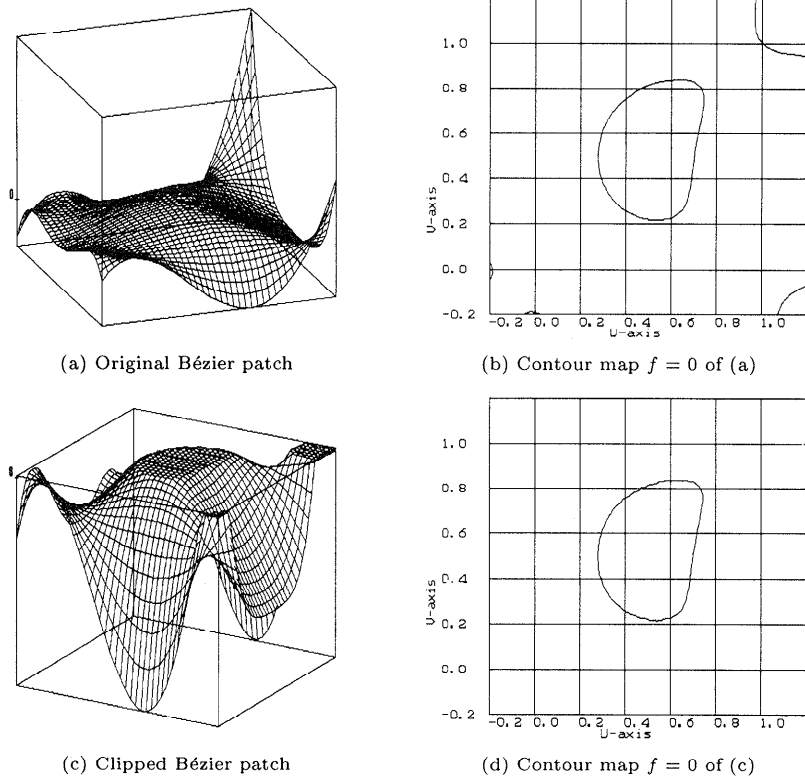


図2 関数クリッピング操作

Fig. 2 Functional clipping operation.

と定義される。

この操作を適用した例を図2(c)と(d)に示す。この方法で単位正方形内の曲線の形状を変更せずに、その外での関数の値を負に変更することができる。3次元ソリッドの場合も同様に、

$$S_{clip}(u, v, w) = S(u, v, w) \cap F_c(u, v, w) \quad (37)$$

であり、ここで単位立方体 $F_c(u, v, w)$ は、

$$F_c = F_b(u) \cap F_b(v) \cap F_b(w) \quad (38)$$

と定義される。

この操作の実装では関数クリッピングをパラメトリックパッチやボリウムに施している。このことによってこれまでの陰関数表現された立体とパラメトリック関数により定義された立体とを区別することなく処理することができる。

パッチやボリウムの定義域が一般の領域に変更された場合には、まず (x, y) を (u, v) に、あるいは (x, y, z) を (u, v, w) にマッピングを行ってから、関数 $S_{clip}(u, v)$ 、あるいは $S_{clip}(u, v, w)$ を定義関数としてその値を計算する。

3.2 拡張 Bézier クリッピング

ここでは2次元ソリッドを例としてパラメトリック

関数 $S(u, v)$ で定義された関数 $f(x, y)$ の定義域の変更について述べる。 $f(x, y)$ の定義域の変更は、 $S(u, v)$ の定義域は単位正方形のままとするが $S(u, v)$ を変形し、 $(x(u, v), y(u, v))$ の値域を単位正方形からより一般的な領域に変更することを意味し、式(24)や式(32)で表されるように規則的に制御点を配置するのではなく一般的な位置に配置する。ここで述べる方法をより高次の場合に拡張するのは容易である。定義域は複雑になるが、 (x, y) から (u, v) へのマッピングの後で関数クリッピングを行うことに困難はない。

制御点の x, y 座標値には任意の値を設定できるので、4本の Bézier 曲線で囲まれた一般的な領域を設定することもできる。図3はそのような一般的な定義域の変更の例を示す。これは図1(a)に示したパッチの各制御点の z 座標値を変更せずに変換したものである。これは1つの変形手法と考えることもできる。

一般的な領域を設定した場合、 P_{ij}^x, P_{ij}^y は一般的な位置にあるので、 $f(x, y)$ の値を求めるには、 $(x, y) = (x(u, v), y(u, v))$ となる (u, v) を求める必要がある。与えられた点 (x_0, y_0) でのパラメータ値 (u, v) を得るために、点 $(x_0, y_0, 0)$ を通過し z 軸に平行な直線

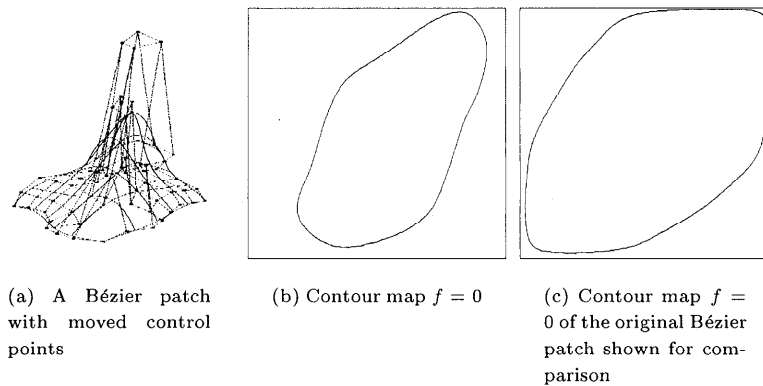


図3 一般的な定義域

Fig. 3 General domain case.

とパッチ $S(u, v)$ との交点を計算しなければならない。Nishita ら³⁾によって提案された Bézier クリッピング法のように、Bézier パッチと直線との交点の算出にはいろいろな方法が提案されている。この論文では交点の計算時間を短縮するために、近似逆マッピングと Bézier クリッピングを組み合わせた拡張 Bézier クリッピングを提案する。この手法は、近似逆マッピングにより交点の存在する可能性の有無を判定し、可能性のある場合に Bézier クリッピングを適用し交点を算出する。拡張 Bézier クリッピングは以下の2つのステップから構成される。

- (1) 与えられた点 (x_0, y_0) に近似逆マッピングを施す。このマッピングはグリーン関数に基づくボリュームスプラインによるまばらなデータの内挿法⁷⁾を用いる。2次元の場合、ボリュームスプラインは薄板スプラインとなることが知られている。このスプラインを使って与えられた点の変位を近似的に計算する。
- (2) 前ステップで計算された (u, v) の値が単位正方形の中に存在すれば、その点を取り囲む境界箱を定義し Bézier クリッピングを実行する。

この方法ではどの点に対しても逆マッピングを行うことができ、表1に示すように純粋な Bézier クリッピングよりも高速である。

図4は純粋な Bézier クリッピングと拡張クリッピングのクリッピングの繰返し数を比較したものである。比較は図3に示したパッチに対して行っている。(b)はパッチを z 軸方向から眺めた図を制御点を結ぶ線とともに描いている。(c)と(d)を比較すると(c)が明るい緑色から青くなっている(繰返し数は9~11回)のに対して(d)の方が暗く赤みを帯び(繰返し数は1

表1 Indy MIPS R4600 64 M バイトによる図3(a)に示したパッチに900点交点計算を行った処理時間の比較

Table 1 Comparison of processing time with 900 points for the patch shown in Fig. 3(a) by Indy MIPS R4600 64 Mbyte.

Algorithm	Tolerance of parameters	
	10^{-6}	10^{-10}
Extended Bézier clipping (using 16 points)	3.49 sec	4.11 sec
(using 25 points)	3.57 sec	4.18 sec
(using 36 points)	3.67 sec	4.27 sec
Pure Bézier clipping	4.84 sec	5.71 sec

~6回)、逆マッピングによって繰返し数が減少していることが分かる。

図5は図4に用いた例の誤差を比較したものである。(b)と(c)を比較すると両者とも同じような色を示しており、誤差がほぼ同程度あることが分かる。

4. パラメトリックパッチとボリュームによるモデリング

27個の制御点を持つ $2 \times 2 \times 2$ 次 (tri-quadratic) Bézier ボリュームと関数クリッピングを使った自由曲面プリミティブを図6(a)に示す。以下にこのプリミティブに対するいくつかの操作を示す。各々の操作の結果、新たに3つの変数で与えられる連続した実数値関数として立体が表現されていることに注意する。立体のレンダリングはユーザが各光線のステップ量を指示できる ray-marching アルゴリズムによって生成している。

4.1 集合演算

関数クリッピングを施した後であれば、自由曲面プリミティブは R 関数を用いて集合演算することが

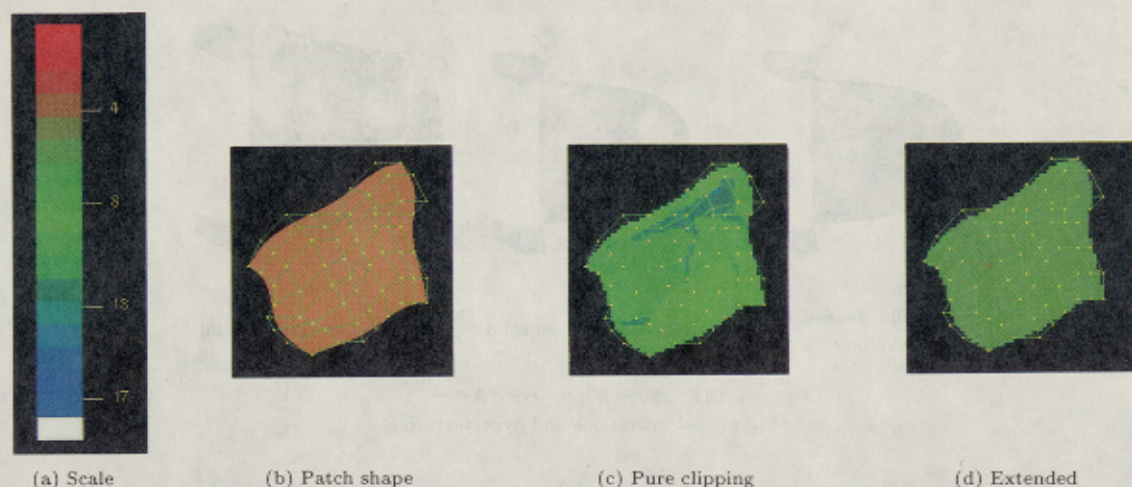


図4 クリッピング回数の比較 (トレランス = 10^{-6} , 逆マッピングには16点使用)
Fig. 4 Comparison of iteration numbers of clippings (tolerance = 10^{-6} , using 16 points for inverse mapping).

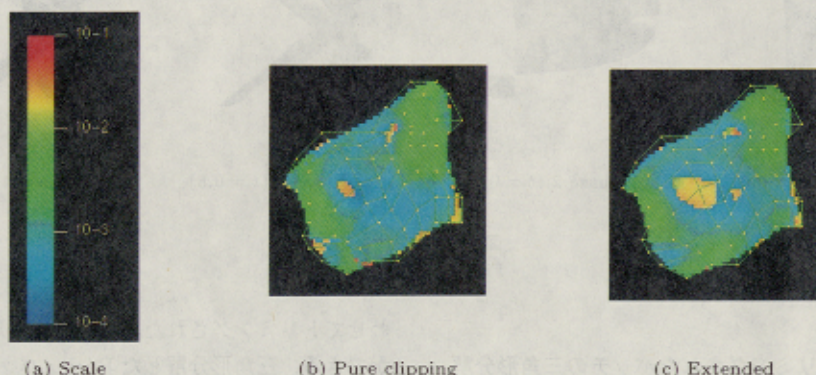


図5 誤差の比較 (トレランス = 10^{-6} , 逆マッピングには16点使用)
Fig. 5 Comparison of errors of clippings (tolerance = 10^{-6} , using 16 points for inverse mapping).

できる。図6(b)は集合演算の結果得られる。これらの演算の詳細は文献4), 9)に述べられている。R関数を用いた集合演算の結果得られるソリッドは正則(regular)であるとは保証されない。低次元のダングリング(dangling)部分を持つ場合や内部の0点を持つ場合がある。実用的な応用では正則であることが要求される場合もあり、その場合には定義関数を再定義する(たとえば文献9))必要がある。

4.2 オフセットとハイパテクスチャ

図6(c)はオフセットとハイパテクスチャ操作の例である。細い針はソリッドノイズ(solid noise)で生成し、オフセットによってカットしている⁴⁾。オフセットは、

$$f_2(x, y, z) = f_1(x, y, z) + d(x, y, z) \quad (39)$$

により実行している。ここで f_1 は初期立体の定義関

数であり、 d は正の値を持つ連続変位関数である。最も単純な例として、 $d(x, y, z) = \text{定数}$ の場合には一定値オフセット(iso-valued offsetting)となる。

4.3 モーフィング

図7は $f_1(x, y, z)$ と $f_2(x, y, z)$ の関数で定義される2つの立体の間のモーフィングを示している。中間の立体は、

$$f_3(x, y, z, t) = (1-t)f_1(x, y, z)h_1(x, y, z, t) + tf_2(x, y, z)h_2(x, y, z, t) \quad (40)$$

によって定義される。ここでパラメータ t は0から1へ変化し、 h_1 と h_2 は正の値を持つ連続モジュレーション関数である。最も単純な場合は $h_1 = h_2 = 1$ である。この例では f_1 は図6(b)に示した自由曲面プリミティブを用い、 f_2 には図7(d)に示した伝統的な手法を用いて二次曲面と超二次曲面によって定義し

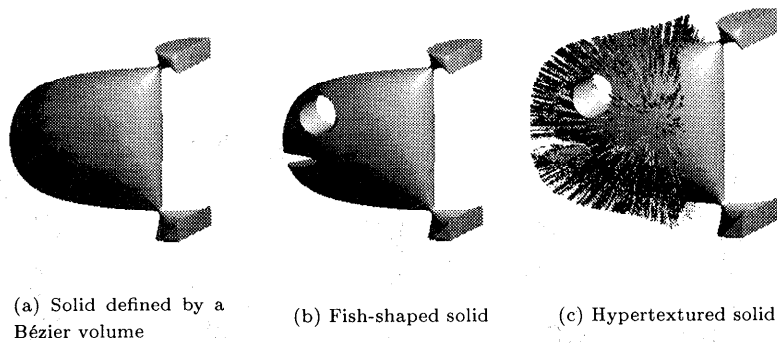


図6 集合演算とハイパテクスチャ
Fig. 6 Set operations and hypertexturing.

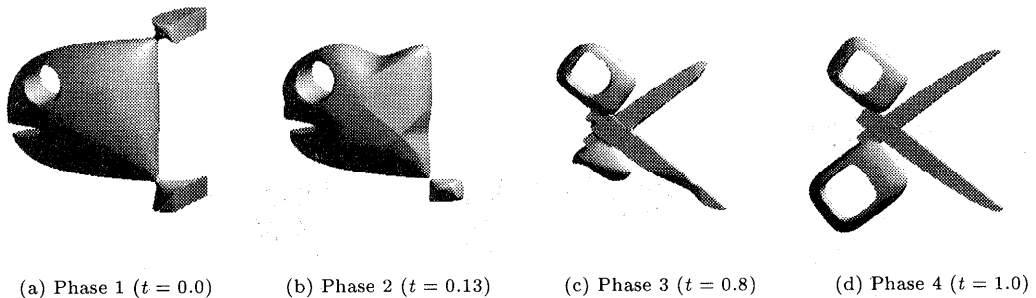


図7 モーフィング
Fig. 7 Morphing.

た立体を用いた。

4.4 応用: トリミングされたパッチの三角形分解

CAGD (computer aided geometric design) では、トリミングされたパッチは重要であり、ここでは本論文で提案した手法の応用例としてその三角形分解 (tessellation) について考える。トリミングされたパッチは以下のように定義される。

- (1) パラメトリックパッチ (ここでは図8に示した Bézier パッチ)
- (2) パッチのパラメータ空間内に定義されたトリミング曲線 (図8(b))。

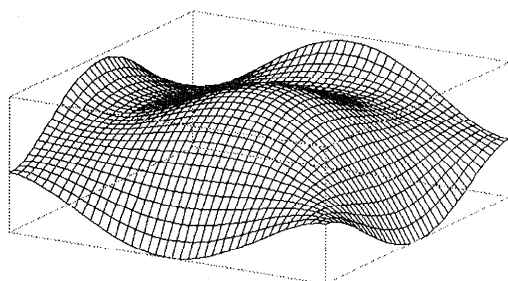
トリミング曲線は別の Bézier パッチの 0 点の集合として定義する。トリミングをパラメータの単位正方形とトリミング曲線で囲まれた領域の「積」の集合演算と考えることができる。トリミング曲線が複数個ある場合にはこの「積」の集合演算を繰り返して施せばよい。この操作の結果、穴が空いていたり、角が落とされているような 2 次元領域が得られる。この領域の三角形分解は、通常陰関数曲面三角形化プログラム (surface tiler) によって行うことができる。それに合

わせてトリミングされたパッチを三角形分解することができる。三角形分解した Bézier パッチを図8(c)に示す。

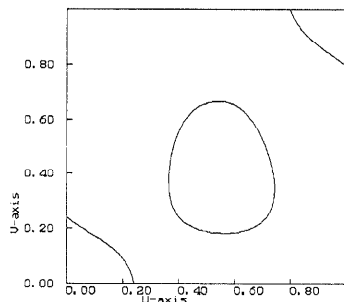
5. おわりに

本研究ではパラメータを用いて定義される自由曲面パッチとボリウムにより陰関数表現によってソリッドを定義する新しい手法を提案した。陰関数表現は立体の定義に広く使われているが、その変形は容易ではない。これを克服するために、パラメトリックパッチとボリウムをこの論文で導入した関数クリッピング操作と拡張 Bézier クリッピング操作ともに用いた。提案する手法を用いれば、R 関数に基づく集合演算においてこれまでの陰関数表現されたソリッドと同じように取り扱うことができる。ソリッドを陰関数表現で表現したり、モデリングを行うためにパラメトリック関数を用いる基本的な取扱いについて説明した。

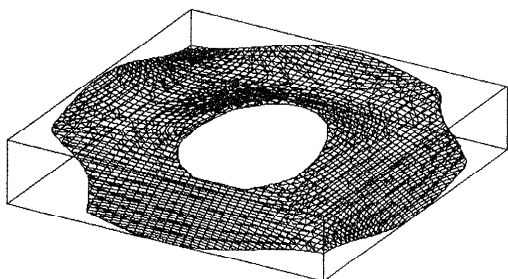
謝辞 アルゴリズムの実装等手伝ってくれた会津大学の Pasko, A.A., Savchenko, V.V. 両氏に感謝する。



(a) Original Bézier patch



(b) Trimming curves in the parameter space



(c) Trimmed Bézier patch

図8 トリミングしたパッチの三角形分解
Fig. 8 Tessellation of trimmed patch.

参考文献

- 1) Menon, J.P.: Constructive Shell Representations for Free-form Surfaces and Solids, *IEEE Computer Graphics and Applications*, Vol.14, No.2, pp.24-27 (1994).
- 2) Middleditch, A.E. and Dimas, E.: Solid Models with Piecewise Algebraic Free-form Faces, CSG94 Set-theoretic Solid Modelling: Tech-

niques and Applications, *Information Geometers*, Winchester, pp.133-148 (1994).

- 3) Nishita, T., Sederberg, T.W. and Kakimoto, M.: Ray Tracing Trimmed Rational Surfaces Patches, *Computer Graphics*, Vol.24, No.3, pp.337-345 (1990).
- 4) Pasko, A., Adzhiev, V., Sourin, A. and Savchenko, V.: Function Representation in Geometric Modeling: Concepts, Implementation and Applications, *Visual Computer*, Vol.11, No.8, pp.429-446 (1995).
- 5) Patricialakis, N.M. and Kriezis, G.A.: Piecewise Continuous Algebraic Surfaces in Terms of B-splines, *Geometric Modeling for Product Engineering*, Wozny, M.J., et al. (Eds.), pp.3-19, Elsevier Science Publishers (1990).
- 6) Rvachev, V.L.: *Methods of Logic Algebra in Mathematical Physics*, Naukova Dumka Publishers, Kiev, Ukraine (1974).
- 7) Savchenko, V.V., Pasko, A.A., Okunev, O.G. and Kunii, T.L.: Function Representation of Solids Reconstructed from Scattered Surface Points and Contours, *Computer Graphics Forum*, Vol.14, No.4, pp.181-188 (1995).
- 8) Sederberg, T.W.: Piecewise Algebraic Surface patches, *Computer Aided Geometric Design*, Vol.2, pp.53-59 (1985).
- 9) Shapiro, V.: Real Functions for Representation of Rigid Solids, *Computer Aided Geometric Design*, Vol.11, No.2, pp.152-175 (1994).

(平成 8 年 8 月 5 日受付)

(平成 9 年 2 月 5 日採録)

三浦憲二郎 (正会員)



昭和 34 年生。昭和 57 年東京大学工学部精密機械工学科卒業。昭和 59 年同大学院修士課程修了。同年、キャノン(株)入社。機械系 CAD/CAM システムの開発に従事。平成 3 年コーネル大学機械工学科博士課程修了。平成 5 年会津大学コンピュータ理工学部コンピュータソフトウェア学科助教授。Ph.D. 曲面の設計, CAD/CAM, 要素自動分割, マイクロマシン等に興味を持つ。ACM, ASME 各会員。