形状モデラのための基本インタフェース*

小 林 一 也** 山 口 泰*** 木 村 文 彦†

Basic Interface for Geometric Modeling Systems

Kazuya KOBAYASHI, Yasushi YAMAGUCHI and Fumihiko KIMURA

Geometric modeling systems must have a variety of capabilities to describe products' shapes, because machine functionalities are realized by its geometric features. However, conventional geometric modeling systems are not powerful and flexible enough for handling those features of products in design and manufacturing processes. In order to cope with this problem, basic interface for geometric modeling systems using generalized topology and geometry is proposed in this paper. The interface provides functionalities for handling several types of topology models, such as solids, surfaces, and wireframe edges. A non-manifold data structure based on neighborhood relationships is used for integrating those topology models. The non-manifold representation is sufficient and efficient for describing adjacency ordering and is also suitable for operations managing connectivities. Since geometric information is generalized by abstracting geometric data type, the interface is so flexible to adapt itself to specific geometric data. Furthermore, the interface enables us to handle simultaneously various types of geometric data by using multiple descriptions.

Key word : CAD/CAM, geometric modeling, wireframe model, surface model, solid model, non-manifold model, Euler operations

1. はじめに

製品形状を計算機上で表現していく上で、様々な問題領域に適した表現で形状を扱うことは重要である。例えば、形状のおおまかな作りをワイヤフレームモデルで表現したり、形状表面にだけ注目してサーフェスモデルを用いたりすることは、問題を簡単化し、設計者にとって見通しの良いシステムとなる。最終的に製品形状はソリッドモデルとして表現する必要がある、各種の形状表現からソリッドモデルに変換することで多くの有用なアプリケーション群を適用できる。

このような個別目的の形状モデラを、場面に応じて容易に実現するためには、幾何や位相の計算で処理内容が共通するものに対して適当なインタフェースを設定し、必要な機能を組み合わせてモデラを構成する方法が考えられる。こうして組み上げたモデラは共通の

データ操作に基づいているので、異なる表現形式の形状を扱うように構成したモデラの間でもデータ交換が可能となる。例えば、形状の表示や入力、計算誤差の制御などの処理を、ワイヤフレームモデラやソリッドモデラといった別構成のモデラ間で、共通に利用できるようになる。

本研究の目的は形状モデラの基礎的な処理について基本インタフェースを設定し、様々な形状モデラの構成を容易にすることである。これには次の利点がある

- ・様々な形状モデラの開発コストを圧縮できる。
- ・扱う形状表現が個々のモデラ間で異なっても、モデルデータを共通に管理できる。

このような基本インタフェースを設定するためには一般性の高い操作を位相・幾何について実現し提供する必要があります。

本論文では、まず基本インタフェースの全体構成について述べ、位相インタフェース・幾何インタフェースのそれぞれについて述べ、最後にまとめを行う。

2. 基本インタフェースの構成

基本インタフェースは、基礎部分(Basic Part)を最下

* 原稿受付 平成3年5月27日

** 正会員 富山県立大学工学部(富山県射水郡小杉町黒河)

*** 正会員 東京電機大学工学部(東京都千代田区神田錦町2-2)

† 正会員 東京大学工学部(東京都文京区本郷7-3-1)

Fig.1 Structure of basic interface for shape modeling systems

層にして、位相インタフェース(Topology Interface)と幾何インタフェース(Geometry Interface)の2つから構成される(図1)。この上に個別目的の形状モデラが構築され、データ操作・基本計算・幾何操作・位相操作など必要な処理を呼び出すことで、モデラに要求される機能を達成できるようにインタフェースを設定する。

処理を位相と幾何に分類した理由は、位相と幾何とは扱う対象の広がり異なるためである。位相インタフェースに関しては、3次元形状を表現するのに十分な“非多様体モデル”を基本とし、その上で個別モデラとのインタフェースを提供する。一方、幾何インタフェースでは“一般幾何”という抽象概念をもとに、任意の種類の幾何要素が扱えるインタフェースを提供する。つまり位相インタフェースでは、3次元の位相という前提で個別対象のインタフェースを構成するのに対して、幾何インタフェースでは自由な拡張を許す構成としている。そこで、基本インタフェースの構成を容易にするために、幾何と位相を分離している。

各部分ごとにその機能を概観する。

2.1 基礎部分の構成

基本インタフェースの最下層には、処理系に依存する部分として、“データ定義(Data Definition)”, “基本計算(Basic Calculation)”, “ポインタ操作(Pointer Management)”がある。データ定義では、幾何処理や位相操作の処理単位となる「要素型」を定義し、要素型に対応するメモリの管理を関数を通じて行う。これにより、異なる構成をとった形状モデラ間での共通のデータ参照が可能になる。基本計算では、 3×1 や 3×3 のマトリクス型の数値計算(和・積・逆行列など)と、数値誤差を統一的に扱うための誤差判定の関数を、距

離・正規化量(基準値 1.0)・角度の各々に対して用意する。ポインタ操作は、位相データ構造の処理を行い、位相要素間のリング構造や親子関係などのポインタ参照を管理する。

これらを基礎にして、上位の位相操作の実現を容易にする“隣接位相操作(Neighborhood Operation)”と、直線や平面といった具体的個別的对象を処理する“個別対象の幾何操作(Specific Geometric Operation)”が構成される。

2.2 位相インタフェースの構成

位相操作部分は、“基本オイラー操作(Basic Euler Operation)”, “個別モデルのオイラー操作(Euler Operation for Specific Model)”, そして“幾何判定を含むオイラー操作(Euler Operation with Geometry)”から成る。基本オイラー操作は、非多様体モデルのオイラー操作で

ある。個別モデルのオイラー操作は、ソリッド/サーフェス/ワイヤフレームなどのクラスに応じたオイラー操作である。幾何判定を含むオイラー操作は、より上位の形状操作を支援するため、幾何情報を利用した位相操作を提供する。例えば、頂点から稜線を延ばす操作を行うとき、稜線の幾何情報を用いて、稜線を延ばす側の判定を内部で行う。

2.3 幾何インタフェースの構成

幾何インタフェースで扱う幾何要素は、点・曲線・曲面の3種類に分類される。特に、一般化した幾何要素を一般点・一般曲線・一般曲面と呼び、総称して一般幾何要素(General Geometric Element)と定義する。幾何インタフェースは一般幾何要素を処理対象とし、“一般幾何操作(General Geometric Operation)”, “交差計算(Intersection Operation)”, “位相を考慮した交差計算(Intersection Operation with Topology)”から成る。一般幾何操作および交差計算は、“個別対象の幾何操作”の処理対象を、一般幾何要素に変更したインタフェースである。位相を考慮した交差計算では、交差の対象として位相要素を取り、計算結果も位相要素となる。例えば、面と面の交差計算の結果は、稜線および頂点となる。

3. 位相インタフェース

前述のように、本研究の位相インタフェースではソリッドモデルに限定せず、より一般的な位相情報を扱う。つまり、正規化集合演算に関して閉じることや、単一の3次元閉領域のみでなくワイヤフレームやサーフェスを扱うことなどを可能とする。さらに非多様体

モデルを基礎として、様々な応用に即した個別の位相インタフェースを提供する。

3.1 3次元非多様体位相

非多様体モデルに関しては、近年盛んに研究が進められている¹⁾。非多様体モデルはソリッドモデルと同様にVertex(頂点), Edge(稜線), Face(面), Region(領域)といった点集合とその隣接関係を表現するが、ソリッドモデルとの相違は隣接順序関係の複雑さである。複数の要素から成り循環する隣接順序関係を、「サイクル」とよぶ。ソリッドモデルでは2種類のサイクルが存在するのに対し、3次元非多様体モデルでは3種類(Loop, Radial, Disk)のサイクルの存在が示されている²⁾³⁾。LoopサイクルはFaceまわりでのEdgeとVertexの隣接順序関係を表す。RadialサイクルはEdgeまわりでのFaceとRegion, DiskサイクルはVertexまわりでのEdgeとFaceの隣接順序関係である。つまり、データ構造の観点からは隣接関係の表現が複雑になるし、データ操作の観点からは隣接関係の処理が複雑化する。

このような非多様体位相の隣接関係の取扱いに有効な概念として隣接位相要素を提案した⁴⁾。隣接位相要素は、点集合間の隣接関係を明示的に表現する仮想的な位相要素であり、データ構造ならびにデータ操作を整理するうえで非常に有用である。隣接位相要素には、隣接する点集合の組合せに応じて、End [VertexとEdge], Fan [VertexとFace], Corner [VertexとRegion], Blade [EdgeとFace], Wedge [EdgeとRegion], Side [FaceとRegion]の6種類がある(図2)。

3.2 データ構造

実際の位相データ構造は図3のようになっている。ここではFan, Blade, Wedgeといった隣接位相要素に代わり、隣接位相要素Featherを用いている。Featherは、Fanの片面, Wedgeの一端, Bladeの半分にあたる位相要素であり、2個1組のFeatherでFan, Blade, Wedgeを表現する。図4は、Featherにより隣接関係を表現した例である。

3種類のサイクルなどの隣接関係はFeather間のポイントによって表現される。本研究では、FeatherはFan, Blade, Wedge各組の相手(Mate)を指すポイントをもつ。この3つのポイントはFan, Blade, Wedgeを明示的に表すものである。具体的に図4の例では、Feather A2とC4がWedge-Mateポイントによって互いに指し合う関係になる。これら3つのMateポイントは、3種類のサイクルを効率良く表現し、プログラミングにも有利であることが分かっている⁴⁾。

3.3 データ操作

位相データの操作に関して、ソリッドモデルでは、

Fig.2 Entities for representing adjacency

Fig.3 Data structure for topology

Fig.4 Feathers

データ構造の完備性を保証するオイラー操作が知られている⁵⁾⁶⁾。オイラー操作は、オイラー式という位相

要素数の間の制約に応じて、位相要素を生成・消去するものである。本研究では、式(1)を用いて図5のようなオイラー操作を実現した。

$$N\{Vertex\} - N\{Edge\} + (N\{Face\} - N\{A.Loop\}) = N\{A.Shell\} - N\{C.Cycle\} + N\{Region\} \quad (1)$$

ここで、 $N\{\}$ は位相要素の数であり、 $A.Loop$ は $Face$ の 2 番目以降の $Loop$ 、 $A.Shell$ は $Region$ の 2 番目以降の $Shell$ を意味する。 $C.Cycle$ は 2 胞複体上で 1 点に収縮できないカットサイクルを表している。図5では、オイラー操作の名前を省略形で表している。大文字は基本的な位相要素 (Region, Shell, ..., Verex) を、小文字は生成 (make) と消去 (kill) を表す。例えば、 $mSkL$ は「 $Shell$ の生成と $Loop$ の消去」($make_Shell_kill_Loop$) を意味する。

一般にオイラー式は基本的な位相要素間の制約式であり、隣接関係を表現する付加的な隣接位相要素の生成・消去については何の指標も与えない。この問題を解決するのが隣接位相操作で、以下の制約式に基づいている。

$$N\{Side\} = 2N\{Face\} \quad (2)$$

$$N\{Wedge\} = N\{Blade\} \quad (3)$$

$$N\{End\} - N\{Fan\} + (N\{Corner\} - N\{A.Disk\}) = 2N\{Vertex\} \quad (4)$$

上記の3式は、それぞれ $Face$ 、 $Edge$ 、 $Vertex$ まわりでの、隣接位相要素の個数に関する制約を表している。

この隣接位相操作を用いることによって、オイラー操作が比較的容易に実現できる⁴⁾。隣接位相操作は基本インタフェースの基礎部分に位置づけられ、位相インタフェースとして提供されるのはオイラー操作より上位のものである。そのため、隣接位相操作はアプリケーションからは隠される。

3.4 個別インタフェース

上記の操作は、非多様体データとしての整合性を保証するが、ワイヤフレームやサーフェス、ソリッドとして閉じたものではない。そこで、非多様体モデル用のオイラー操作の一部を制限したり、適当に組み合わせることによって、ワイヤフレーム、サーフェス、ソリッドとして閉じた位相操作を提供する。例えばワイヤフレームは頂点と稜線のみで構成されるため、 $Face$ 、 $Loop$ 、 $Region$ といった位相要素は不要である。基本位相操作から、 $Face$ 、 $Loop$ 、 $Region$ に関する操作を除外したものが、ワイヤフレームの位相操作となる。

サーフェスとして整合性と完備性を備えた位相操作を実現するためには、基本位相操作を制限するだけでなく、適当に組み合わせる必要がある。例えば mVS

Fig.5 Euler operations

Fig.6 Euler operations for solid model

や kEF は、サーフェスモデルの範囲を外れた位相データを生成してしまう。そこで、これらを組み合わせた $mEVFS$ をサーフェスモデル用のオイラー操作として提供する。

ソリッドの場合でも同様に、 $Region$ に関する操作の mFR は、実行結果を非多様体形状にしてしまう可能性がある。そこで mFR を $mSkL$ と組み合わせて1つの操作である $mFSRkL$ とすれば、ソリッドとして閉じた操作になる。こうして提供される操作は、図6に示すように、ソリッドモデルのオイラー操作として良く知られたものと等価である。

4. 幾何インタフェース

4.1 一般幾何要素

すでに述べたように、幾何インタフェースでは一般幾何要素を処理の対象にとる。直線や平面といった具体的な幾何情報は、一般幾何要素の属性情報として保持される。

一般幾何の導入により、様々な幾何要素固有の特徴を隠蔽し、上位の応用から幾何操作が行いやすくなる。また、基本インタフェース全体の維持・管理も容易に

なる。例えば、2 曲面間の交線計算を考える。これは一般幾何を対象とするので、入力に2つの一般曲面をとり、結果として複数の一般曲線と一般点を出力する。これにより、平面と平面、平面と球面、といった曲面の種類による分類を、操作を呼び出す側で意識する必要がなくなる。また、一般幾何を通じて幾何操作を記述する限り、位相操作や、より上位の応用に影響を及ぼさずに、幾何要素の種類を拡張できる。

4.2 幾何的な特徴量とパラメータ化

幾何インタフェースでは、幾何要素の「特徴」を得るための操作を提供している。ここでいう「特徴」とは幾何的な諸量で表現され、座標値、1階/2階偏微分値、パラメータ、パラメータ化情報のそれぞれを扱う。必要なパラメータの個数は一般幾何要素の次元数に等しい。表1に、幾何的な「特徴」を扱う一般幾何操作を示す。ここで $\text{func}(in \downarrow, out \uparrow)$ とは、 func という名前の操作が、引数 in を入力に out を出力に取ることを意味する。

$\text{point_curve_surface_coordinate}$ の各操作は、一般幾何要素についてパラメータ値から3次元直交座標値を求める。この逆操作が $\text{curve_surface_parameter}$ であり、座標値からパラメータを計算する。ただし座標値が曲線・曲面の上にないと、得られるパラメータ値は誤差を含みうる。 $\text{curve_surface_derivative}$ は、1階/2階偏微分値を計算する。ただし混合偏微分の値が微分の順序により異なる場合は扱わない。これらの操作を元にして、曲線の方角ベクトルを計算する操作 curve_direction と、曲面の法線ベクトルを求める操作 surface_normal 、曲面の曲率を求める操作 surface_curvature を実現する。

パラメータ化の方法は、幾何要素の種類やパラメータ範囲によって異なる。例えば、パラメータを用いて曲線上をたどる場合、曲線が円であるなら、曲線の形は閉じていてパラメータ範囲は $(-\pi, \pi]$ であるが、直線なら開いていて範囲は $(-\infty, \infty)$ である。この違いが認識できないと、首尾よく曲線の追跡を行えない。すなわち一般幾何要素のパラメータ化には、幾何要素の形やパラメータ範囲(これらを“パラメータ化情報”と定義する)が必要となる。 curve_param_info は、一般曲線のパラメータ化情報を求める操作である。曲線の形は、閉じているか開いているかの2通りに分類できる。パラメータ範囲は有限な値で通常表されるが、曲線が開いている場合は $\pm\infty$ となる場合も含む。 $\text{surface_param_info}$ は、矩形状にパラメータ化される一般曲面に限り、パラメータ化情報を u/v 方向別に与える。 point_param_info は、一般点の座標

Table 1 General geometric operations for computing geometric features

t : curve parameter / u, v : surface parameters
 $in \downarrow$: input argument / $out \uparrow$: output argument

point_coordinates	$(\text{point} \downarrow, \text{coord} \uparrow)$
point_param_info	$(\text{point} \downarrow, \text{kind} \uparrow)$
curve_parameter	$(\text{curve} \downarrow, \text{coord} \downarrow, t \uparrow)$
curve_coordinates	$(\text{curve} \downarrow, t \downarrow, \text{coord} \uparrow)$
curve_direction	$(\text{curve} \downarrow, t \downarrow, \text{dir} \uparrow)$
curve_curvature	$(\text{curve} \downarrow, t \downarrow, \text{crvt} \uparrow, \text{rad} \uparrow)$
curve_derivative	$(\text{curve} \downarrow, t \downarrow, \text{deg_t} \downarrow, \text{deriv} \uparrow)$
curve_param_info	$(\text{curve} \downarrow, \text{shape_t} \uparrow, \text{min_t} \uparrow, \text{max_t} \uparrow)$
surface_parameter	$(\text{surface} \downarrow, \text{coord} \downarrow, u \uparrow, v \uparrow)$
$\text{surface_coordinates}$	$(\text{surface} \downarrow, u \downarrow, v \downarrow, \text{coord} \uparrow)$
surface_normal	$(\text{surface} \downarrow, u \downarrow, v \downarrow, \text{nrml} \uparrow)$
surface_curvature	$(\text{surface} \downarrow, u \downarrow, v \downarrow, \text{crvt_l} \uparrow, \text{crvt_t} \uparrow, \text{rad_l} \uparrow, \text{rad_t} \uparrow)$
$\text{surface_derivative}$	$(\text{surface} \downarrow, \text{deg_u} \downarrow, \text{deg_v} \downarrow, u \downarrow, v \downarrow, \text{deriv} \uparrow)$
$\text{surface_param_info}$	$(\text{surface} \downarrow, \text{shape_u} \uparrow, \text{min_u} \uparrow, \text{max_u} \uparrow, \text{shape_v} \uparrow, \text{min_v} \uparrow, \text{max_v} \uparrow)$

Table 2 Structures of general geometric entities

Point (relation with Vertex)	
primal data:	<i>Cartesian_coordinates, Polar_coordinates</i>
source data:	<i>Point_on_curve, Point_on_surface</i>
auxiliary data:	<i>Point_with_normal</i>
Curve (relation with Edge)	
primal data:	<i>Line, Circle, Ellipse, Parabola, Hyperbola, Bézier_curve</i>
source data:	<i>Curve_on_surface</i> <i>(Bézier_curve_2D, Polyline_2D)</i>
auxiliary data:	<i>Polyline</i>
Surface (relation with Face)	
primal data:	<i>Plane, Sphere, Cylinder, Cone, Torus, Bézier_surface</i>
source data:	<i>Surface_interpolation</i>
auxiliary data:	<i>Triangle_patches, Polyhedron</i>

値の表現形式が、直交座標系、極座標系、円柱座標系のいずれであるかを与える。

4.3 一般幾何要素による多重表現

一般幾何要素は表2に示すように、複数の表現を同時に持つことができ、主表現 (primal data)、元表現 (source data)、補助表現 (auxiliary data) に分類されている。一般幾何要素が保持できる幾何要素の個数は、主表現では1つ、元表現では0以上、補助表現では0

または1となっている。多重表現を採用した理由は、処理に応じて最も適した表現を保持しておき、計算時間の節約や数値計算誤差を小さくできるからである。これは幾何操作で普遍的に行われる「点と領域の包含判定」などに活用できる。

主表現は、一般幾何要素の「主たる」幾何情報であり、幾何インタフェースにおいて座標値や交線の計算というように、新しい要素を導出する形式の幾何操作から参照される。例えば曲面の主表現は、平面(Plane)、球面(Sphere)、円柱(Cylinder)、円錐(Cone)、トーラス(Torus)、Bézier曲面(Bézier_surface)のどれか1種類をとる。

元表現は、一般幾何の主表現を補佐する幾何情報である。幾何操作の結果として一般幾何要素が生成されるとき、主表現の「起源」の情報を保持する幾何要素が、元表現として付け加えられる。例えば、曲面間の交線計算では交線は一般曲線となるが、その元表現は曲面上曲線(Curve_on_surface)となる。曲面上曲線は、参照曲面の他に、2次元Bézier曲線(Bézier_curve_2D)または2次元折れ線(Polyline_2D)で表される。そのため主表現の変更が起きると、元表現は消去または更新される。元表現はパラメータ値の計算や包含判定のように、判定を行う形式の操作から参照される。

補助表現は、主表現から生成できるので、常に保持する必要はない。補助表現は、陰面処理などの高速な表示や、正確さよりも高い応答性が要求される応用で用いられる。ハードウェアの現状を考えると、多角形(Polyline)や多面体(Polyhedron)がこうした応用に向いていると考えられる。一般幾何要素を適当なパラメータ刻み幅に分割すると、一般曲線の補助表現は多角形となり、一般曲面では多面体となる。補助表現は主表現が変更されるまでは再計算せず、応答性の高いモデリングシステムを支援できる。

様々な幾何要素を統合して扱うために、幾何情報を一般化した一般幾何要素の概念を導入した。一般幾何要素を用いると、幾何的な「特徴」としての諸量は一般幾何操作を通じて統一的に扱うことができる。一般幾何の情報は多重表現が可能のように構造化されており、主たる単一の表現と、補助的な複数の表現を持ち、処理に適した表現で幾何情報を保持する。

5. ま と め

形状モデラを構成するために基本インタフェースを設定することで、以下の点が明らかとなった。

- ・モデリングシステムの作成・保守が容易になる。
- ・非多様体モデルを位相構造の核とすることで、さまざまな位相を有する形状に対応できる。
- ・一般幾何要素の導入により、さまざまな幾何要素を統一的に扱うことができ、幾何情報の多重表現が可能となる。

応用例として曲面モデラの試作を行ったが、詳細は文献⁷⁾⁸⁾を参照されたい。この実装はC言語で行い、“データ定義”での要素型をC言語の「型」に、各インタフェースにおける“操作”をC言語の「関数」に対応させた。そのためメモリーを操作する操作名が要素型により異なるなど、不都合な点も見られた。そこで将来は、関数の呼び出しをより柔軟に行うために、C++言語による実装を考えている。

謝 辞

本研究の一部は精密工学会産学協同研究協議会“製品の機能や生産性の事前評価にもとづく機械生産の高精度化研究協力分科会”によるものである。

参 考 文 献

- 1) K.J. Weiler: Topological Structures for Geometric Modeling, Ph.D. Thesis of Rensselaer Polytechnic Institute, (1986).
- 2) E.L. Gursoz, Y. Choi & F.B. Prinz: Vertex-based Representation of Non-manifold Boundaries, Geometric Modeling for Product Engineering, North-Holland, Netherland (1990) 107.
- 3) E. Brisson: Representing Geometric Structures in d Dimensions: Topology and Order, ACM Symp. on Computational Geometry, (1989) 218.
- 4) 山口 泰, 木村文彦: 非多様体位相の隣接関係の表現と操作, 情報処理学会論文誌, 32, 6 (1991) 731.
- 5) I.C. Braid, R.C. Hillyard & I.A. Stroud: Stepwise Construction of Polyhedra in Geometric Modelling, Mathematical Method in Computer Graphics and Design, Academic Press, (1980) 123.
- 6) M. Mäntylä: An Introduction to Solid Modeling, Computer Science Press, U.S.A. (1988) 401.
- 7) 中島 洋, 木村文彦, 山口 泰, 小林一也: 有理曲面式による自由曲面形状の生成と評価, 情報処理学会「グラフィックスとCAD」シンポジウム, (1989) 37.
- 8) 中島 洋: 有理ベジエ式による曲線と曲面の生成, 1989年度東京大学大学院修士論文, (1990).