401　高速積層造形法における三角形パッチ幾何データフォーマット

北海道大学工学部　○平澤友康　田中文基　岸浪建史

要旨

高速積層造形装置への最も一般的な入力幾何データであるSTLデータは、位相構造を持たない独立した三角形の集合で立体の幾何形状を表している。このため、成型物加工の際、様々な問題が生じている。本研究では、STLデータの問題を解決するため、STEPのFacetted B-rep(ISO 10303-204)に基づいた幾何形状入力データフォーマットの構造を提案する。

1.はじめに

STLデータフォーマットは、高速積層造形装置への入力幾何データとして最も一般的なもので、成型物の表面形状の、三角形パッチ（3つの頂点と法線で表す）の集合による近似表現である[1]。しかし、STLデータフォーマットには、データが冗長なる、モデルの正当性を自己検証することができない、といった問題がある。

本研究では、このような問題を解決するために、STEP Facetted B-rep(ISO 10303-204)[2]に基づいた、位相構造を持ち、冗長さのない入力幾何データフォーマットの構造を提案する。

2.STLデータフォーマットの問題点

STLデータフォーマットの問題点として、以下の2点が挙げられる（図1参照）。

1.データが冗長になる。

各三角形パッチを独立に定義するために、複数の三角形によって共有される節点が、それを共有する三角形ごとに何度も定義される。このため、必要以上にデータ量が大きくなってしまう。

2.立体の正当性の自己検証機能がない。

STLデータは、面の接続情報を持たないため、面同士の接続を保証できない。このため、CADデータからSTLデータへの変換の際、三角形間の隙間や重複、立体の内外を示す法線ベクトルの不統一などの誤差が生じ、成型できない場合がある。

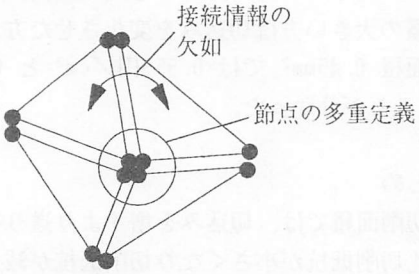


図1 STLデータフォーマットの問題点

3.新フォーマットの提案

3.1基本的考え方

新フォーマット提案に対する基本的考え方を図2に示す。高速積層造形装置への入力データの満たすべき条件として、1.高速スライス生成機能（高速積層造形装置でのデータのスライスが可能なこと）、2.CADシステムとの適合性、が挙げられる。また、STLデータフォーマットの問題点を解決する条件としては、3.冗長なデータの削除、4.自己検証機能、が挙げられる。これらの条件から新フォーマットには以下のような要求事項が導かれる。

○スライス平面と立体表面の交線を高速に求めることを可能とするため、立体表面を三角形平面で近似する。

○CADシステムへの実装を可能にするため、製品データの記述・交換のためのISO規格であるSTEP(STandard for the Exchange of Product data)を利用する。

○冗長なデータを削減するため、不必要な要素を省いた効率的なデータ記述フォーマットとする。

○立体の正当性の自己検証を可能にするため、データは要素同士の隣接関係を表現できる位相構造を持つ必要がある。

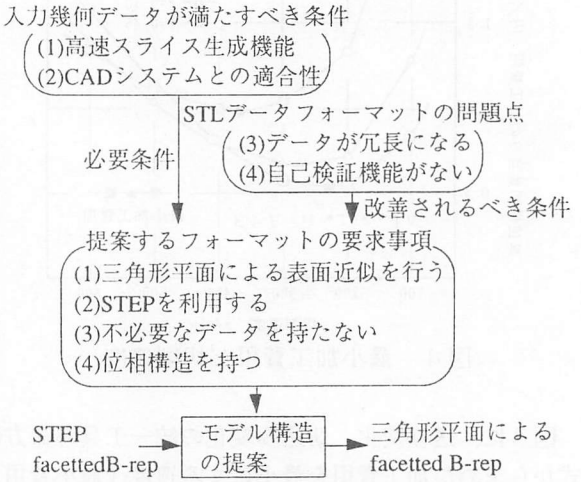


図2.新フォーマット提案の基本的考え方

3.2新フォーマットの構造

上記の要求事項から、STEP Part204のFacetted B-repに基づき、facetの形状を三角形に限定した、三角形Facetted B-repフォーマットを提案する。

幾何形状を表現する要素以外は入力幾何データとしては不必要なので、STEP Facetted B-repの全ての要素の中から、表現対象、座標系、マッピング、座標変換に関する部分を削除した。さらにfacetは穴を許容しないので、面中の穴に関する要素を削除し、それに応じて他の要素を修正した。また、冗長な記述を避けるために不必要な（導出可能な）幾何要素を削除した。

図3は、提案するフォーマットの構造をEXPRESS-G表現したものである。facetted_brepは、closed_shellを属性（outer）として持つsolid_modelの下位型として定義される。位相情報は、poly_loop（loopの下位型）、face、closed_shell（connected_face_setの下位型）で扱う。幾何情報は、cartesian_pointとpoly_loopで表現し、poly_loopは、辺が直線であることを暗に定義する。poly_loopの属性であるpolygonを要素数3のcartesian_pointのリストとしていることで、facetの形状を三角形に限定している。

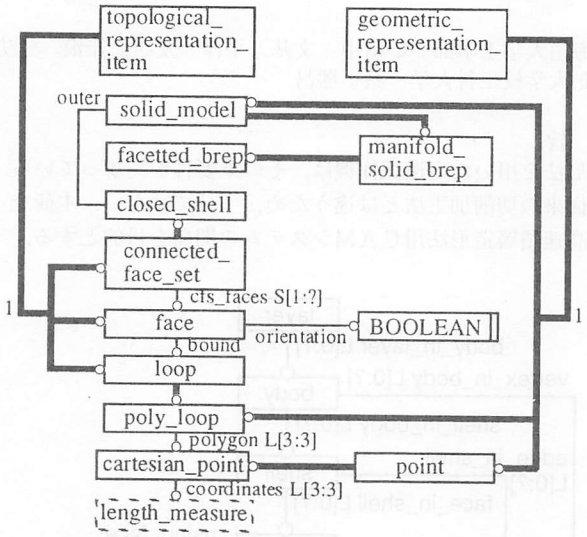


図3.三角形facetted Brepモデルの構造

4.処理系と実装例

4.1処理系の構造

提案したフォーマットの有効性の検証を目的として、STLファイルから提案したフォーマットのファイルへ変換する処理系を作成した。図4にそのアルゴリズムを示す。まず、STLデータを三角形パッチごとに読み込み、節点のcartesian_pointへの登録、または登録済みcartesian_pointの呼び出しを行い、poly_loopに登録する。さらに生成されたpoly_loopからfaceを生成し、closed_shellに登録する。この操作を繰り返し、全てのfaceを登録したclosed_shellをfacetted_brepに登録する。

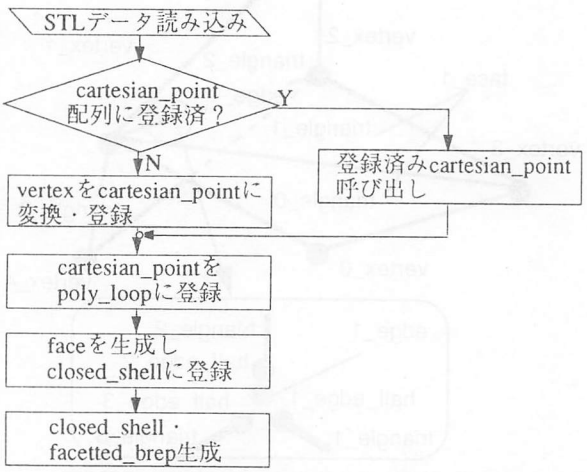
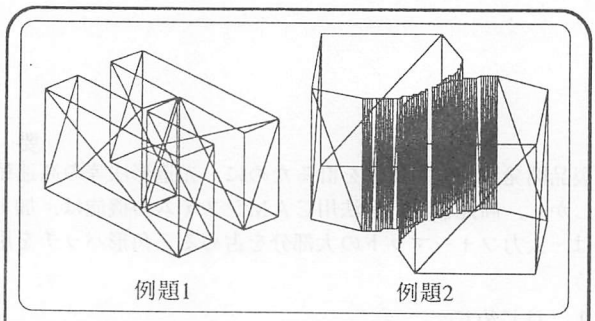


図4.処理系のアルゴリズム

4.2実行例と比較

例題と実行結果を図5に、また、STLファイルと三角形Facetted B-repファイルの比較を表1に示す。STLファイルでは三角形節点を多重定義しているが、三角形Facetted B-repファイルでは、一度定義した要素が再び使用される場合は、ポインタ形式で指定する。また、STLファイルは位相要素を持たないが、三角形Facetted B-repファイルは持つ。例題1、2ともファイルサイズは約60％に縮小された。

以上の比較結果より、提案したフォーマットの有効性が実証された。




```
solid
 facet normal 1.000000 -0.000000 0.000000
   outer loop
     vertex 50.000000 30.000000 30.000000
     vertex 50.000000 20.000000 30.000000
     vertex 50.000000 20.000000 0.000000
   endloop
 endfacet
 . . . . . .
endsolid
```

STLファイル（例題１）

```
#10 = CARTESIAN_POINT ((50., 30., 30.));
#20 = CARTESIAN_POINT ((50., 20., 30.));
#30 = CARTESIAN_POINT ((50., 20., 0.));
#40 = POLY_LOOP ((#10, #20, #30));
#50 = FACE (#40, .T.);
#60 = CARTESIAN_POINT ((50., 30., 0.));
#70 = POLY_LOOP ((#30, #60, #10));
#80 = FACE (#70, .T.);
. . . . . .
#730 = CLOSED_SHELL ((#50, #80,・・・#720));
#740 = FACETTED_BREP (#730);
```

三角形facetted B-repファイル（例題１）

図５.実行例

表1.比較

	STLファイル	三角形facettedB-repファイル
ファイルサイズ（バイト）	4988（例題1）	2892
	55773（例題2）	33553
点の定義	三角形ごとに何度も行う	1度（ポインタで指定）
面の定義	ループと法線	ループと方向性を示すBOOLEAN flag
立体の定義	無	閉じたシェル
位相構造	無	有

5.結論

本研究では以下のことを行った。

1.STLデータフォーマットの問題点を指摘した。

2.STEP Facetted B-repをもとにした三角形Facetted B-repフォーマットを提案した。

3.処理系を作成し、新フォーマットの有効性を確認した。

参考文献

[1]Paul F.Jacobs編・著,Thomas H.Pang,嶋田洋訳,"高速3次元形成の基礎",日経BP出版センター

[2]ISO 10303-204,"Mechanical design using boundary representation"