1　はじめに

計算機の進歩に伴い、3 次元モデリングの技術や応用分野が急速に拡大し、3 次元物体のモデリングに関する研究が新しい情報科学の基盤の 1 つになってきている [1]。形状モデリングにおいて、形状が複雑かつ設計図がない対象物に対しては、計測したボリュームデータから形状モデリングすることを考える。形状モデリングに良く用いられる 3 次元物体の表現手法としては、ポリゴン表現、パラメトリック曲面表現、大局的多面体表現などがある [2]。その中でも対象物の境界の情報を持ち、さらに形状情報を効率・精度よく抽出できることが期待されている大局的多面体表現を用いてボリュームデータから形状モデリングを行う [3]。(本稿では大局的多面体を略称し多面体と呼ぶこともある。) 多面体表現は多面体化アルゴリズムを用いて形状モデリングを行う。しかし、従来の多面体化アルゴリズムでは、ボリュームデータから多面体を生成する時には、多面体を構成する平面の数、位置、大きさが一通りに決められる方法は与えられていないという問題がある [4]。そこで本稿では、ボリュームデータに対して直接に形状解析を行い、その結果を用いて上記の問題解決を試みる。まず、ボリュームデータのような離散的な点の配置から物体表面の平坦な度合いを表す新しい尺度、平坦度を定義する。さらに、平坦度を求めるアルゴリズムを提案し、実際にボリュームデータに適用した結果を示す。そして、平坦度を用いて上記の問題を解決する多面体化アルゴリズムの検討を行う。

2　多面体化アルゴリズム

3 次元形状の大局的多面体表現とは、与えられた対象物のボリュームデータに対し、多面体化アルゴリズムを用いて、平面の集合である多面体を用いて対象物を形状モデリングすることである。

ここでは、3 次元ユークリッド空間 $\boldsymbol{R}^3$ において座標値が全て整数値である点の集合を $\boldsymbol{Z}^3$ とし、離散空間と呼ぶ。本研究で扱うボリュームデータ $\boldsymbol{V}$ は $\boldsymbol{Z}^3$ の部分集合である。

2.1　離散平面

多面体は平面の集合であり、以下に $\boldsymbol{Z}^3$ における平面を定義する。$\boldsymbol{Z}^3$ では式 (1) を満たすような点 (x, y, z) の集合を離散平面という [5]。

$$0 \leq ax + by + cz + d < w \tag{1}$$

但し、全パラメータは整数である。(a, b, c) は離散平面の法線ベクトルであり、a, b, c, d は離散平面の係数と呼ぶ。$w > 0$ であり、離散平面の幅という。w によって平面の種類が決定される。$w = \max(|a|, |b|, |c|)$ の時、離散平面を特に naive plane という [6]。今後、naive plane を略称し平面と呼ぶ。

2.2　多面体化アルゴリズム

対象物のボリュームデータ $\boldsymbol{V} \subset \boldsymbol{Z}^3$ からその境界部分の点集合 $\boldsymbol{B} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ は、文献 [6] のアルゴリズムによって一意に求められ、$\boldsymbol{B} \subset \boldsymbol{V} \subset \boldsymbol{Z}^3$ となる。また、文献 [7] により平面によって分けられる $\boldsymbol{Z}^3$ の部分集合に [6] のアルゴリズムを適用した結果、得られる $\boldsymbol{B}$ は、式 (1) から求められた点の集合と等しいことが証明されている。

離散平面の有限部分集合 $\boldsymbol{P}_i$ を平面分と呼ぶと、$\boldsymbol{P}_i \subset \boldsymbol{B}$、かつ $\cup_{i=1}^{m} \boldsymbol{P}_i = \boldsymbol{B}$ となる複数の $\boldsymbol{P}_i$ を考えることができる。各 $\boldsymbol{P}_i$ は式 (1) を満たす点の部分集合なので、平面の係数の組 (a_i, b_i, c_i, d_i) で表すこともできる。そして、多面体は平面分の和集合 $\cup_{i=1}^{m} \boldsymbol{P}_i$ で表される。m, n は整数である。多面体化アルゴリズムは $\boldsymbol{B}$ を $\cup_{i=1}^{m} \boldsymbol{P}_i$ に分解するアルゴリズムである。多面体化アルゴリズムには、点の集合 $\boldsymbol{Q}$ に対し、平面分であるかどうか判定する関数 $planarity(\boldsymbol{Q})$ が必要となる。

$planarity(\boldsymbol{Q}) =$

$$\begin{cases} 1 & \boldsymbol{Q}\text{には式 (1) を満たさない点がない} \\ 0 & \boldsymbol{Q}\text{には式 (1) を満たさない点がある} \end{cases} \tag{2}$$

なお、計算機上で式 (2) の実現には、文献 [8] の平面判定アルゴリズムを用いる。

多面体化アルゴリズムを以下に示す。

```
Algorithm1
Input: B = {v_1, ..., v_n}
Output: ∪_{i=1}^{m} P_i
begin
1. i := 1;
2. while(B ≠ φ ) {
   2.1. Q ⊂ B を選ぶ;
   2.2. while( planarity(Q)==1) {
            2.2.1. v ∈ B \ Q を選ぶ;
            2.2.2. Q: = Q ∪ { v };
     }
   2.3. Q := Q \ { v };
   2.4. P_i := Q;
   2.5. B := B\ Q ;
   2.6. i + +;
}
end
```

なお、ステップ 2.4 において $\boldsymbol{P}_i$ に対応する部分集合 $\boldsymbol{Q}$ を $\boldsymbol{Q}_i$ とも書く。

2.3 多面体化アルゴリズムの問題点

前章で説明した多面体化アルゴリズムは、得られた多面体の平面の数、位置、大きさがアルゴリズム 1 のステップ 2.1 の部分集合 $\boldsymbol{Q}$ の選択及びステップ 2.2 の while 文内の $\boldsymbol{Q}$ の拡張方法によって変わってくる。即ち、$\boldsymbol{Q}$ の選択・拡張方法によって異なった平面分 $\boldsymbol{P}_i$ が得られてしまう。この問題を解決するためには部分集合 $\boldsymbol{Q}$ の上手な選択・拡張方法を考える必要がある。現在のところ、$\boldsymbol{B}$ から一意に $\boldsymbol{Q}$ を選ぶ方法は提案されていない [4]。

3 平坦度によるデータの形状解析

多面体化アルゴリズムの問題を解決するために、ボリュームデータ点の平坦度を新たに定義する。平坦度を用いて、多面体化アルゴリズムの部分集合 $\boldsymbol{Q}$ の選択・拡張方法に制約をつけることで、良い $\boldsymbol{P}_i$ を得るような部分集合 $\boldsymbol{Q}$ の選択・拡張方法を決めることを考える。なお、具体的な $\boldsymbol{Q}$ の選択・拡張の仕方については 4 章で述べる。

3.1 平坦度の定義

多面体化アルゴリズムでは、点の集合 $\boldsymbol{Q}$ に対し、平面分 $(planarity)$ であるかどうかの判定を行なっている。一般的に知られている $\boldsymbol{Z}^3$ における $planarity$ の定義は式 (2) のように与えられている [5, 8, 9]。

点の集合 $\boldsymbol{Q}$ に対する平面分であるかどうかの判定 $planarity(\boldsymbol{Q})$ に対し、本研究では、各点 $\boldsymbol{v}$ の周りの点の分布から $\boldsymbol{v}$ の平坦の度合いを表す平坦度 $(flatness)$ を定義する。

$$flatness(\boldsymbol{v}) = N(\boldsymbol{v}) \geq 3 \tag{3}$$

ここで、$N(\boldsymbol{v})$ は $\boldsymbol{v}$ の周りの平面である条件を満たす点の集合の広がりを表し、平面の大きさや面積に対応する。即ち、平坦度は $\boldsymbol{v}$ 及びその周りの点の集合が平面を満たすかどうかだけではなく、平面である条件を満たす場合、平面の広がりという平坦の度合い (平面の可能性) を表すものである。但し、実際に 1 つの平面を決定するのに 3 つ以上の点が必要なので、$N(\boldsymbol{v}) \geq 3$ とする。

また、ボリュームデータの解像度や対象物の大きさによって、平坦度の値が異なり、異なる物体間での平坦度の相対関係を導き出すことができず、一概に全対象物体の平坦度の最大値を定めることができない。そのため、現段階では、平坦度の正規化を行なわない。単一物体のボリュームデータに対して、平坦度の絶対値ではなく、相対的な大小関係を表す相対値のみを扱う。

平坦度を用いた部分集合 $\boldsymbol{Q}$ の選択方法を考えると、平坦度の大きい点ほど周りの点の集合が平面である可能性が高いため、部分集合 $\boldsymbol{Q}$ の選択には平坦度の高い点を優先することが挙げられる。

3.2 平坦度算出アルゴリズム

点 $\boldsymbol{v} \in \boldsymbol{B}$ $(\boldsymbol{B} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\})$、における平坦度 $flatness(\boldsymbol{v})$ を求めるには、点 $\boldsymbol{v}$ の対象集合 $\boldsymbol{O}_j(\boldsymbol{v})$ が必要となる (詳しくは 3.3.2 節参照)。

$$\boldsymbol{O}_j(\boldsymbol{v}) = \{\boldsymbol{v}, \boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_j\} \tag{4}$$

但し、$0 \leq j \leq n$、$\boldsymbol{O}_0(\boldsymbol{v}) = \{\ \boldsymbol{v}\ \}$、$\boldsymbol{O}_j(\boldsymbol{v}) \subseteq \boldsymbol{B}$ である。$\boldsymbol{O}_j(\boldsymbol{v})$ を求めるのに順序付けアルゴリズムを用いる (3.3.2 節参照)。

点 $\boldsymbol{v}$ の平坦度 $flatness(\boldsymbol{v})$ を以下のアルゴリズムによって求める。

Algorithm2
Input: $v \in B$
Output: $flatness(v)$
begin
1. $j := 2$;
2. while($planarity(O_j(v))==1$) {
   2.1. $j++$;
   }
3. $j--$;
4. $flatness(v) := |O_j(v)|$;
end

なお、$|\boldsymbol{O}_j(\boldsymbol{v})|$ は $\boldsymbol{O}_j(\boldsymbol{v})$ が含む点の数である。

3.3 三角形分割構造を用いた点順序付けアルゴリズム

アルゴリズム 2 のステップ 2 における $\boldsymbol{v}$ の対象集合 $\boldsymbol{O}_j(\boldsymbol{v})$ を決定するアルゴリズムを検討し、順序付けアルゴリズムと呼ぶ。ここで、文献 [6] のアルゴリズムにより求めたボリュームデータ $\boldsymbol{V}$ の境界部分集合 $\boldsymbol{B}$ は三角形分割されている。順序付けアルゴリズムには、三角形分割構造に基づき定義された星状体や星状体の縁などを用いる。まず、星状体などの定義について説明する。

3.3.1 離散平面の星状体

星状体などの定義を説明するのに、離散複体などの知識が必要となる。それらを以下に説明する。また、文献 [7] により、本稿で扱う naive plane の三角形分割は 18,26 近傍の離散単体による三角形分割になる。

なお、$\boldsymbol{v} = (x, y, z) \in \boldsymbol{Z}^3$ に対し、$N_{18}(\boldsymbol{v}) = \{(p, q, r) \mid (x-p)^2 + (y-q)^2 + (z-r)^2 \geq 2\}$ を満たすものを $\boldsymbol{v}$ の 18 近傍と呼び [10]、$N_{26}(\boldsymbol{v}) = \{(p, q, r) \mid (x-p)^2 + (y-q)^2 + (z-r)^2 \geq 3\}$ を満たすものを $\boldsymbol{v}$ の 26 近傍と

呼ぶ [10]。但し、$p, q, r \in \boldsymbol{Z}$ である。また、定義される離散 (0,1,2 次元) 単体は図 1 のようになる (構成方法は [10] 参照)。横軸は 18,26 近傍であり、縦軸は 0,1,2 次元である。

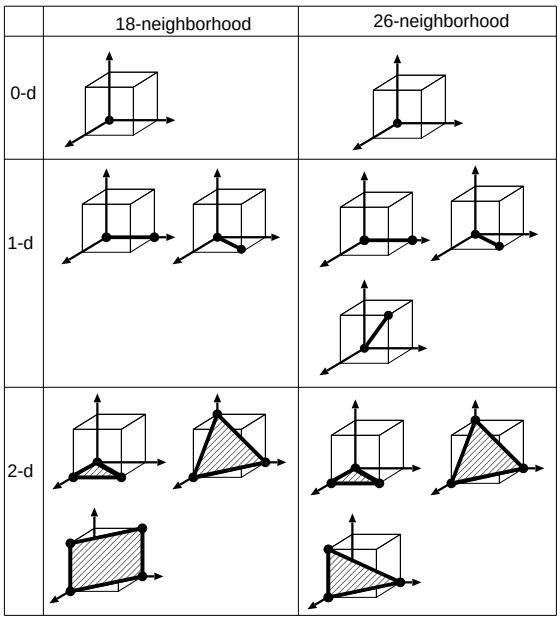

図 1: 離散単体

定義 1 離散複体 *[10]*

離散単体の有限集合 $\boldsymbol{K}$ が以下の条件を満たす場合、$\boldsymbol{K}$ を離散複体、または略して複体と呼ぶ。

(1) $[a] \in \boldsymbol{K}$ ならば、$face([a]) \subseteq \boldsymbol{K}$

(2) $[a_1], [a_2] \in \boldsymbol{K}$, $\mid a_1 \mid \cap \mid a_2 \mid \neq \phi$ ならば、$[a_1]$=$[a_2]$

$\boldsymbol{K}$ に含むすべての離散単体の最大次元数を $\boldsymbol{K}$ の次元とする。なお、$[a]$ は離散 (r 次元) 単体であり、$[a]$=$[\boldsymbol{v}_0$,$\boldsymbol{v}_1$, $\cdots$,$\boldsymbol{v}$, $\cdots$,$\boldsymbol{v}_k]$ とも書く。$\boldsymbol{v}_0$,$\boldsymbol{v}_1$, $\cdots$, $\boldsymbol{v}$,$\cdots$,$\boldsymbol{v}_k$ は $[a]$ の頂点と呼ぶ。$face([a])$ は $[a]$ の部分集合である離散 s(次元) 単体 $(s < r)$ の和集合であり、$[a]$ の辺集合ともよぶ。$\mid a \mid$=$\mid \boldsymbol{v}_0, \boldsymbol{v}_1, \cdots, \boldsymbol{v}, \cdots, \boldsymbol{v}_k \mid = \{\boldsymbol{v} \mid \sum_{i=0}^{k} t_i \boldsymbol{v}_i, \sum_{i=0}^{k} t_i = 1, t_i > 0\}$ であり、$[a]$ の埋め込み単体とも呼ぶ。

ボリュームデータ $\boldsymbol{V}$ の三角形分割される境界部分集合 $\boldsymbol{B}$ は離散複体である。

定義 2 星状体 *[10]*

$\boldsymbol{v} \in \boldsymbol{K}$ の星状体 $\boldsymbol{S}(\boldsymbol{v})$ は以下のように定義する。

$$\boldsymbol{S}(\boldsymbol{v}) = \{[a] \mid \boldsymbol{v} \in [a], [a] \in \boldsymbol{K}\} \tag{5}$$

定義 3 星状体の縁 *[10]*

点 $\boldsymbol{v}$ の星状体の縁 $\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$ は以下のように定義する。

$$\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor = \cup_{[a] \in \boldsymbol{S}(\boldsymbol{v})} face([a]) \setminus \boldsymbol{S}(\boldsymbol{v}) \tag{6}$$

定義 4 輪状星状体 *[10]*

点 $\boldsymbol{v}$ の星状体の縁 $\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$ が以下の条件を満たす時、$\boldsymbol{S}(\boldsymbol{v})$ を輪状星状体という。そうでなければ非輪状星状体と呼ぶ。

(1) $\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$ は *1* 次元離散複体である

(2) $[a] \in \lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$ ならば、$[a] \in face([b])$

(但し : $[a]$ は離散 r (次元) 単体であり $(r < 1)$、$[b]$ は $\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$ の離散 *1* (次元) 単体の 1 つである)

(3) $\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$ の二つの要素 $[a]$ と $[b]$ に対し、$[a_1]$ = $[a]$,$[a_2]$,$\cdots$, $[a_k]$=$[b]$ のような離散単体の行列を見つけることができる

(但し : $[a_1] \in \lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$, $[a_i] \cap [a_{i+1}] \neq \phi$)

(4) $\boldsymbol{v}$ が $\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$ の頂点ならば、$\boldsymbol{S}(\boldsymbol{v}) = \{[\boldsymbol{v}_1], [\boldsymbol{v}_2]\}$

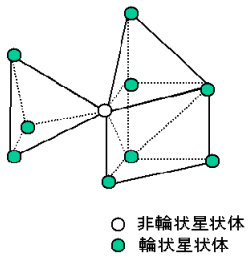

図 2: 星状体の例 (26 近傍)

また、本稿では輪状星状体のみを扱う。輪状星状体の縁を用いてアルゴリズム 2 のステップ 2 の $\boldsymbol{O}_j(\boldsymbol{v})$ を決定する順序付けアルゴリズムを考える。

3.3.2 順序付けアルゴリズム

輪状星状体をもつ点 $\boldsymbol{v}$ に対し、$\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$ から始点 $\boldsymbol{v}_1$ を選び、$\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$ の点らを反時計回りすることで、$\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$ 中の点を一意に順序付けすることができる。順序付けアルゴリズムに使われるいくつかの演算や関数を以下に示す。

- $\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor = \{\boldsymbol{v}, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_k\}$ の点を $\boldsymbol{v}_1$ を始点として順序付けする関数

  $Around(\boldsymbol{v}, \boldsymbol{v}_1) = (\boldsymbol{v}_1, \cdots, \boldsymbol{v}_k)$

- 2 つの順序付けられた点列 $b = (\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_i)$ と $b\prime = (\boldsymbol{v}_i, \cdots, \boldsymbol{v}_j)$ の足し算

  $b + b\prime = (\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_i, \boldsymbol{v}_{i+1}, \cdots, \boldsymbol{v}_j)$

- 点列 $b = (\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_j)$ の最後の点を返す関数

  $last(b) = \boldsymbol{v}_j$

- 点列 $b = (\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_{i-1}, \boldsymbol{v}_i, \cdots, \boldsymbol{v}_j)$ のある点 $\boldsymbol{v}_i$ 以降の点列を削除する関数

  $cut(b, \boldsymbol{v}_i) = (\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_{i-1})$

この関数らを用いて、ある対象点 $\boldsymbol{v} \in \boldsymbol{B}$ $(\boldsymbol{B} = \{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n\})$ の対象集合 $\boldsymbol{O}_j(\boldsymbol{v})$ を求める順序付けアルゴリズムを以下に記す。

Algorithm3
Input: v,j
Output: $Que(v) = (v, v_1, \cdots, v_j)$
begin
1. $Que(v) := (v)$;
2. $Que(v) := Que(v) + Around(v\ ,\ v_1)$;
3. $i := 1$;
4. while($|\ Que(v)\ | < j$){
   4.1. $Que(v) := cut(Que(v), last(Que(v))) +$
   $cut(Around(v_i, last(Que(v))), v_{i+1})$;
   }
end

なお、ステップ 2 において、現段階では $\boldsymbol{v}_1$ を点の座標情報 (x, y, z) に基づいて選ぶ。具体的には、$\lfloor \boldsymbol{S}(\boldsymbol{v}) \rfloor$ から z が最大である点、あるいは z が最大の点らから y が最大である点、あるいは z が最大かつ y が最大の点らから x が最大である点を選ぶ。ステップ 4 において、$|\ \boldsymbol{Que}(\boldsymbol{v})\ |$ は $\boldsymbol{Que}(\boldsymbol{v})$ に含む点の数を表す。また、$\boldsymbol{Que}(\boldsymbol{v})$ を $\boldsymbol{O}_j(\boldsymbol{v})$ と呼ぶ。

3.4 平坦度を求める実験

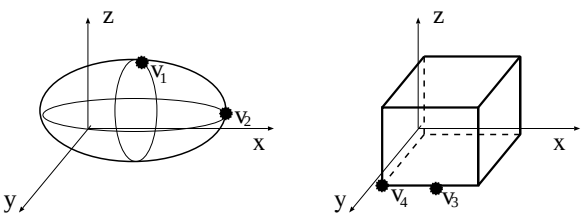

図 3: 実験対象図 楕円体 (左) 及び 長方形 (右)

平坦度の妥当性を確認するために、図 3 に示す楕円体および長方形ボリュームデータのいくつか点 $\boldsymbol{v}_1, \boldsymbol{v}_2$ $\boldsymbol{v}_3, \boldsymbol{v}_4$ に対し、平坦度をそれぞれ求めた。なお、楕円体のボリュームデータは $x^2/10^2 + y^2/15^2 + z^2/8^2 <= 1$ を満たす点 $v(x, y, z)$, 長方形のボリュームデータは $0 < x < 11$,$0 < y < 10$,$0 < z < 9$ を満たす点 $v(x, y, z) \in \boldsymbol{Z}^3$ の集合とした。それぞれの値は $flatness(\boldsymbol{v}_1) = 17, flatness(\boldsymbol{v}_2) = 5$, $flatness(\boldsymbol{v}_3) = 7, flatness(\boldsymbol{v}_4) = 4$ となり、$\boldsymbol{v}_2$ よりも $\boldsymbol{v}_1$ の平坦度が高く、$\boldsymbol{v}_4$ よりも $\boldsymbol{v}_3$ の平坦度が高いという予測と一致した。

4 平坦度を用いた多面体化アルゴリズムの検討

ボリュームデータから多面体を作成する時には、対象物が長方形、正方形のような多面体である場合とそうでない場合では、作成する多面体が異なってくる。従って、平坦度を用いた多面体化アルゴリズムも異る。対象物が多面体の場合は、元の多面体と似た多面体を作成することを考える。一方、対象物が多面体でない場合では、まず、対象物に対する最適な多面体を検討して、得られた最適な多面体を作成する多面体化アルゴリズムを考える。しかしながら、最適な多面体の議論そのものが複雑になるので、本稿では対象物が多面体の場合について、平坦度を用いた多面体化アルゴリズムを検討する。

平坦度を用いてアルゴリズム 1 の $\boldsymbol{Q}$ の選択・拡張方法に制約をつけることによって、$\boldsymbol{Q}$ により得られた多面体が異ってくるという多面体化アルゴリズムの問題点を解決することを考える (2.3 節参照)。

まず、アルゴリズム 1 のステップ 2.1 の $\boldsymbol{Q}$ の選択方法に制約を付けることを考える。平坦度の大きい点ほど周りの点の集合が平面である可能性が高いため、$\boldsymbol{Q}$ の選択には平坦度の高い点を優先することで平面を決定する。即ち、ボリュームデータの平坦度に基づいて、平坦度の高い点から平面分を決定していく。具体的には、アルゴリズム 1 のステップ 2.1 の $\boldsymbol{Q} \subset \boldsymbol{B}$ を選ぶ時には、アルゴリズム 4 のように $\boldsymbol{Q}$ を選択する。但し、$\boldsymbol{O}(\boldsymbol{v})$ は $\boldsymbol{v}$ の平坦度が求められた時の対象集合であり、アルゴリズム 2 のステップ 4 の $\boldsymbol{O}_j(\boldsymbol{v})$ である。

Algorithm4
Input: $B = \{v_1, \cdots, v_n\}$
Output: Q
begin
1. $v \in B$ を選ぶ; $(flatness(v) = \max\{flatness(v_1),$
   $\cdots, flatness(v_n)\})$
2. $Q := O(v)$;
end

次に、アルゴリズム 1 のステップ 2.2 の while 文内の $\boldsymbol{Q}$ の拡張方法に制約を付けることを考えて、アルゴリズムを拡張する。具体的には、アルゴリズム 1 のステップ 2.2. 及び 2.3 はアルゴリズム 5 のように拡張する。なお、アルゴリズム 5 の Input の $\boldsymbol{O}(\boldsymbol{v})$ はアルゴリズム 4 のステップ 2 の $\boldsymbol{O}(\boldsymbol{v})$ であり、アルゴリズム 4 の

Output の Q である。前処理のステップ 0 はすでにアルゴリズム 4 で行なわれたとする。

Algorithm5
Input: $O(v)$
Output: Q
begin
(0. $Q := O(v)$;)
1. for($i =\mid O(v) \mid -1$; $i > 0$;$i--$){
  1.1 $Q := Q \cup growing(O(v), v_i)$;
}
end

なお、ステップ 1.1 の $growing$ 関数は以下のようになり、$growing$ 関数中のステップ 2 の $firstnum(G, \boldsymbol{O}(\boldsymbol{v}))$ は点列 $G = (\boldsymbol{v}_0, \cdots, \boldsymbol{v}_l, \cdots)$ の中で、$\{\boldsymbol{v}_l\} \cup \boldsymbol{O}(\boldsymbol{v})$ が平面判定を満たす最初の点 $\boldsymbol{v}_l \in G$ の番号 l を返す関数である。

function: $growing(O(v), v_i)$
input: $O(v) = (v_0, \cdots, v_i, \cdots)$, $v_i \in O(v)$
output: G
begin
1. $G = cut(Around(v, v_{i+1}), v_{i-1})$;
2. $l := firstnum(G, O(v))$;
3. while($l <\mid G \mid$) {
  3.1. if(($planarity(v_l \cup O(v)) == 1$) && ($v_l \notin O(v)$))
    3.1.1. then {
      3.1.1.1. $O(v) := O(v) \cup \{v_l\}$;
      3.1.1.2. if(($planarity(v_{l+1} \cup O(v)) == 1$) &&
            ($v_{l+1} \notin O(v)$))
        3.1.1.2.1. then {
          3.1.1.2.1.1. $O(v) := O(v) \cup \{v_{l+1}\}$;
          3.1.1.2.1.2. $growing(O(v), v_{l+1})$;
          }
        3.1.1.2.2. else $l := l + 2$;
    3.1.2. else $l := l + 1$;
  }
4. $G := O(v)$;
end

5 まとめ

ボリュームデータを入力とする多面体による形状モデルを実現するために、多面体化アルゴリズムの問題の原因を分析し、問題を解決するために、ボリュームデータに対して平坦度を新たに定義し、解決方法の検討を行なった。さらに、平坦度を求めるアルゴリズムを検討し、実際のデータに適応して実験した。平坦度を用いた多面体化アルゴリズムの検討も行なった。今後の課題として、平面の法線ベクトル、幅、平面の種類を考慮し、平坦度の拡張などがあげられる。

参考文献

[1] H. Zha, T. Mitutomi, T. Hasegawa, "Generation of Multiresolution Model of 3-D Free-From Objects from Range Data",Meeting on Image Recognition and Understanding , vol.2, pp241-246, 2000

[2] 技術編 CG 標準テキストブック編集委員会, "技術編 CG 標準テキストブック", 財団法人画像情報教育振興協会, 1999

[3] R. Klette, J.S. Hao, "A New Global Polyhedrization Algorithm for Digital Regular Solids", to appear

[4] J. Francon, L. Papier, "Polyhedrization of the Boundary of a Voxel Object", Discrete Geometry for Computer Imagery, LNCS 1568,pp425-434,1999

[5] J.P. Reveilles, I.D. Renesson, "Incremental algorithm for recognizing pieces of digital planes", 4^{th} Conference on Discrete Geometry in Computer Imagery,Strasbourg, 1994

[6] Y. Kenmochi, A. Imiya, A. Ichikawa, "Boundary Extraction of Discrete Objects",Computer Vision and Image Understanding,Vol. 71,pp281-293,1998

[7] Y. Kenmochi, A. Imiya, "Naive Planes as Discrete Combinatorial Surfaces", to appear in Proceedings of Discrete Geometry for Computer Imagery

[8] J. Francon, J.M. Schramm, M. Tajine, "Recognizing Arithmetic Straight Lines and Planes", Proc. Int. Workshop on Discrete Geometry for Computer Imagery, LNCS 1176, pp142-149,1996

[9] C.E. Kim, "Three-dimensional digital planes ", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.PAMI-4, pp639-645, 1984

[10] Y. Kenmochi, A. Imiya, A. Ichikawa, "Discrete Combinatorial Geomtetry", Pattern Recognition,Vol. 30,No. 10,pp1719-1728,1997