Runlength を用いた差分流体解析前処理

荒川佳樹† 三谷眞人†† 一柳高時††

Runlength 表現形式を用いて，流体解析における前処理システムを開発した．流体占有率・開口率を用いる差分流体解析では，その前処理において次の三つの主要な課題がある．(1)形状定義モデル (BRep, CSG) からのデータ変換，(2)解析格子の自動生成，(3)流体占有率・開口率の自動計算．ところで，形状モデルは Object Oriented モデルと Space Oriented モデルに大別される．このような前処理では，空間に対するアドレッシング性，ユニーク性に優れた後者が適している．このモデルとしては Voxel と Octree が代表的であるが，Voxel はデータ量，Octree は BRep からのデータ変換性に問題がある．これに対して，Runlength はデータ圧縮性および BRep からのデータ変換性を合わせ持っている．そこで，Runlength を用いて前処理システムを構築し，かつ上記三つの課題を統一的に解決した．また，実際の解析事例を用いて，このシステムの性能評価を行い，その有効性・実用性を実証した．

Pre-processing of FDM-based Numerical Fluid Analysis Using Runlength

Yoshiki Arakawa,† Masato Mitani†† and Takashi Ichiyanagi††

A pre-processor using Runlength is presented for the numerical fluid analysis using fluid occupation ratio and porosity. The following three problems can be distinguished in the pre-processing. Namely, the problems of (1) data conversion from input model (BRep, CSG), (2) automatic grid generation and (3) calculation of fluid occupation ratio and porosity. Space oriented modeling is suitable for the pre-processing because of having the properties of spatial addressing and spatial uniqueness. In the space oriented modeling, it is a common practice to use Voxel or Octree. However, Voxel needs a huge memory and Octree has an inferior ability with respect to the data conversion from BRep. On the contrary, Runlength doesn't have these week points. Therefore, we developed the pre-processor based on Runlength. By doing this, we could solve the above-mentioned three problems.

1. はじめに

近年，図形処理の科学技術分野への適用が非常に盛んになってきているが，流体解析分野などはその代表格である．数値流体解析においては，差分法を始めとして有限要素法など種々の解析手法が用いられるが，**流体占有率・開口率**を用いた差分流体解析における前処理（以下，単に前処理と呼ぶ）では，次の三つの主要な課題がある．

† 郵政省通信総合研究所関西先端研究センター知覚機構研究室
Auditory & Visual Informatics Section, Kansai Advanced Research Center, Communications Research Laboratory, Ministry of Posts & Telecommunications
†† 松下電器産業株式会社生産技術研究所 CAE 研究部
CAE Research Division, Production Engineering Laboratory, Matsushita Electric Industrial Co., Ltd.

（1）形状定義モデルからのデータ変換
（2）解析格子の自動生成
（3）流体占有率・開口率の自動計算

ソリッドモデルは，**Object Oriented** モデルと **Space Oriented** モデルに大別することができる[3]．前者の代表的なものが，**BRep** (**Boundary Representation**) および **CSG** (**Constructive Solid Geometry**) である[3)~5)]．後者に関しては，**Octree**[6)~8)] および **Voxel**[9)] が代表的である．両者は本質的に異なる特性を持っている．すなわち，前者は形状に対するコヒーレンス性，後者は空間に対するアドレッシング性，ユニーク性に優れている[3]．

ところで，流体占有率とは解析セルにおいて形状が占めていない部分の体積比率，開口率とはセル面において形状が占めていない部分の面積比率のことである．したがって，前処理では，解析格子あるいは解析セルと形状との干渉演算が多用される．そこで，この

ような目的に用いるソリッドモデルとしては，Space Oriented モデルが適している．

一方，これは形状モデリング全般に言えることでもあるが，流体解析においても，形状の定義は一般に BRep および CSG が用いられる．これは，これらの Object Oriented モデルの方が対話的に形状定義やその修正が行いやすいこと，形状の原データである設計 (CAD) データは BRep が一般的であることなどによる．しかしながら，前処理において，BRep および CSG の両方のモデルに対応した例は報告されていない．特に，CAD 分野では BRep が主流になってきているので，この BRep データの直接的な利用が望まれている．これが(1)の課題である．

スーパーコンピュータ，超並列計算機などに代表されるように，計算機能力の増大は目覚ましいものがあり，解析対象はより複雑になり，かつ高精度な解析が要求されてきている．また，後処理では，使用されるグラフィック・ディバイスの解像度は今日では 1024^2 レベルが普通であり，将来的にはこれ以上の高精細度かつ大型の表示スクリーンとなっていくであろう．

そこで，形状モデルに要求される表現精度は高くなる一方である．また，解析格子の数は増大の一途をたどり，数十・数百万個のオーダになってきている．このような膨大な格子を適切に生成し，かつ流体占有率・開口率を精度よく計算することには非常な手間を要しており，自動化が望まれている．解析結果を可視化する後処理だけではなく，これらの前処理に対する重要性が一段と増してきている[1]．

これまでに，(3)に関しては，CSG モデルで形状を表現し，空間格子を発生させて，流体占有率・開口率を求める方法が提案されている[2]．しかし，この方法は計算精度を上げようとすると，演算量が膨大となる問題を抱えており，比較的低精度の計算となっている．また，(2)に関しては，任意の形状に対して格子を自動生成する方法は報告されていない．

Space Oriented モデルの中では，Voxel は最も原始的かつ単純なモデルである．これまでは，データ量が膨大になるということであまり評価されてこなかった．しかし，近年の急速な計算機ハードウェア能力の向上により，ボリューム・レンダリングの中核表現として Voxel 表現が非常に注目されてきている．とは言っても，現在の Voxel による実現分解能は高々 512^3 程度である[9]．これを単純に高分解能化していくことはデータ量が膨大となり，現在の計算機技術の延長線上では，依然として非実現的である．

後処理との統合化などを考えても，前処理における形状モデルは 1024^3 以上の高形状表現分解能の実現が必要不可欠であろう．ところで，流体解析において対象となる形状は，自動車，ビルなどの障害物（遮へい物）である．そして，このような形状はそれほど凹凸がなく，何らかのデータ圧縮表現が可能である場合が多い．

このようなデータ圧縮には，Octree が非常に有効であるが，BRep からの変換性で問題がある．例えば，登尾らは多面体表現された BRep から Octree へのデータ変換に取り組んでいるが，その形状は凸でなければならないという制約条件がつく[8]．穴などのある非凸立体に対しては，前処理として凸分割しなければならない．しかし，このような分割を一般的に行うことは難しく，汎用的かつ効率的な方法は知られていない．さらに，近年では曲面を用いた BRep が実用化されつつあるが，このようなモデルから Octree を直接的に生成することはより一層の困難を伴う．

このような観点から，筆者らは Space Oriented モデルに対する見直しを行い，2次元画像処理の分野でよく用いられる **Runlength** 表現の再評価を行ってきている[18),19)]．その結果，Runlength は Object Oriented モデルからのデータ変換性に優れており，かつ比較的高い形状表現分解能（データ圧縮性）を実現できる，というのが筆者らの得た結論である．

そこで，Runlength 表現形式を用いて，上記三つの課題を統一的に解決した前処理システムを構築した．本論文ではこのシステムの概要およびそこで採用されているデータ構造，処理アルゴリズムについて報告する．

まず，2章では，Runlength に関するこれまでの研究をまとめ，本システムの Runlength データ構造について述べる．3章では(1)BRep から Runlength の生成，(2)格子生成，(3)流体占有率・開口率計算の各アルゴリズムについて述べる．4章では，いくつかの実際の解析事例における性能評価結果を報告し，その考察を行う．

2. Runlength モデル

Runlength による形状表現法は，**図1** に示すように，形状を包む直方体空間（以下，形状空間と呼ぶ）を設定し，その一つの表面（ここでは X-Y 平面）を格子分割し，これらの格子点を通りこの面に垂直な直

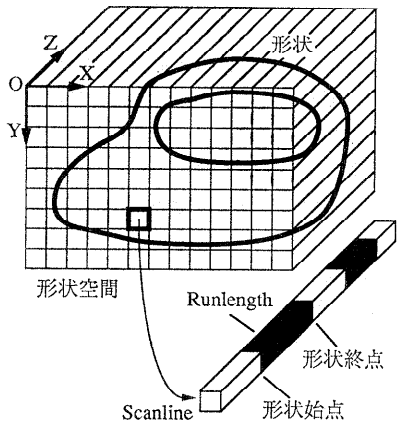


図 1 ランレングス表現
Fig. 1 Runlength representation.

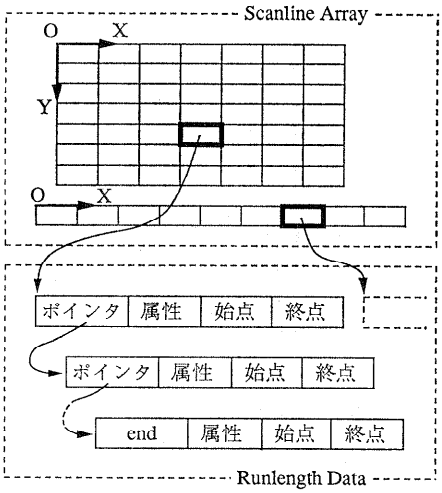


図 2 ランレングスのデータ構造
Fig. 2 Runlength data structure.

線であるスキャンライン（ここではZ軸に平行）を仮想する．これにより，対象形状はスキャンラインと交線，すなわち **Runlength** の集合体で近似表現される．

従来，このような形状表現法は2次元画像処理の分野で Runlength 法として広く用いられてきている．3次元の分野では，画像生成・合成[10)~15)]およびマス・プロパティ計算[15)~17)]などによく用いられてきているが，いずれも限定的あるいは補助的な表現形式に留まっている．そして，この表現形式は，**Runlength**[10)]，**Volume segment**[11)]，**Extended D-buffer**[12),14)]，**Dexel**[13)]，**Ray Casting**[15)]，**Column Decomposition**[17)] などいろいろと呼ばれてきているが，ここでは **Runlength** と呼ぶことにする．

Runlength 表現が3次元の分野において主表現としてあまり用いられないのは，次のような大きな欠点があることによると考えられる．

（1） Octree ほど一般的にデータ圧縮効率がよくなく，座標系に対するユニーク性の欠如が大きい．すなわち，スキャンラインの取り方でデータの内容およびその量が大きく変化する．

（2） スキャンライン方向とそうでない方向に対する処理アルゴリズムが異なり，Octree のように洗練された再帰的なアルゴリズムとならない．

これらの欠点に対する筆者らの見解は次のようである．（1）の欠点に関しては，近年の計算機ハードウェア能力の急速な向上により，Voxel と同様にそれほど問題にならなくなってきている．また，（2）の欠点に関しても，高々数通りの処理アルゴリズムを用意すればよく，いずれも単純なアルゴリズムとなるので，実用上はそれほど問題にならない．

本システムの Runlength データ構造は，**図2**に示すように，スキャンライン配列部と Runlength データ部から構成される．前者は各スキャンラインに対応し，その先頭の Runlength データへのポインタが入っている．そして，3次元形状を表現する2次元配列部と断面図などの2次元形状を表現する1次元配列部から構成される．これにより，同一記憶領域内で，2次元と3次元の両方の形状データが扱える．

また，後者は，1次元固定長リストという非常に単純な構造となる．今回開発したモデラでは，ポインタ4，属性データ4（形状番号2，色番号2），形状始点値2および終点値2の合計 12 バイトから構成される．このように属性データとして，形状番号と色とを合わせ持っているので，複数形状が同一データ領域内で扱え，かつ一つの形状に複数色を持たせることができる．

3. 処理アルゴリズム

3.1 BRep/Runlength 変換

形状の定義には，BRep および CSG モデルが一般的に用いられる．本システムはこれらの両モデルに対応している．CSG モデルに関しては，4面体，3角柱，平行6面体，楕円柱，楕円錐，楕円体，トーラスなどのプリミティブとこれらに対応する Runlength 生成ルーチンおよび形状演算ルーチンを用意して

いる．

BRep モデルからの Runlength 生成は，プリミティブの場合と同様に，境界面とスキャンラインとの交点計算を繰り返せばよいが，この場合は始点値および終点値の両方が同時に確定しない．そこで，Runlength を拡張して，始点値あるいは数点値が未定でもよい Runlength 形式を新たに導入する．

ここでは，BRep モデルは平面多角形面で構成されていると仮定する．この変換処理は次の三つのステップをすべての多角形面（以下，単に多角形と呼ぶ）について繰り返すアルゴリズムとなる．

（ステップ1）

まず，1つの多角形を取り出し，この法線ベクトル $\boldsymbol{n}$ を求める．ここでは形状が存在する側を法線ベクトルの正方向とする．多角形とスキャライン（Z 軸に平行）との交点 z は，$\boldsymbol{n}$ の Z 座標値 n_z が，$n_z>0$ であれば始点，$n_z<0$ であれば終点となる．$n_z=0$ の場合は，多角形とスキャンラインが平行となり交点が存在しないので，次の多角形に処理を進める．

（ステップ2）

次に，この多角形のすべての頂点を調べて，多角形が存在する，X, Y 座標の範囲 $(x_{\min}, y_{\min})-(x_{\max}, y_{\max})$ を求める．そして，この範囲におけるすべてのスキャライン (x, y) $(x_{\min}\leq x\leq x_{\max}, y_{\min}\leq y\leq y_{\max})$ と多角形との交点 z を順次求める．

（ステップ3）

この交点 z と既存する Runlength データ列 (z_{1i}, z_{2i}) $(i=0, 1, 2, \cdots)$ との比較を行う．この比較パターンは**図3**に示すように8通りとなる．図3の(6)を例にとると，z は終点値で，Runlength $(z_{1i}, -)$ と $(z_{1(i+1)}, z_{2(i+1)})$ との間となる場合である．ここで，"−" はその値が未定であることを示す．この間に z を挿入した結果は，(z_{1i}, z), $(z_{1(i+1)}, z_{2(i+1)})$ となる．

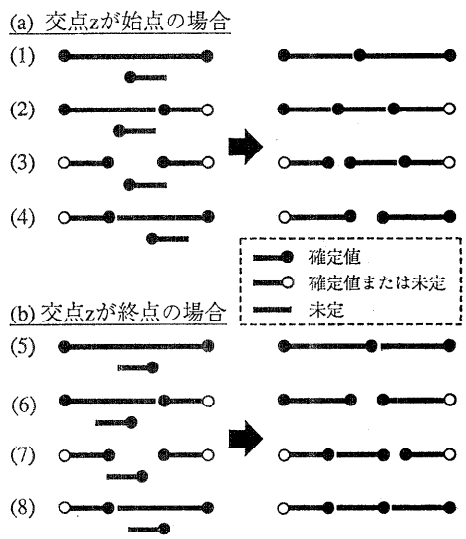


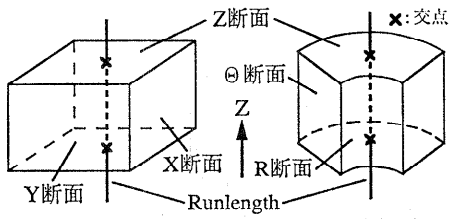
図 3 Runlength の生成
Fig. 3 Runlength generation.

このアルゴリズムでは，面の処理順は任意であり，非凸形状に対しても適用できる．また，曲面の場合は，スキャンラインとの交点およびその点における法線ベクトルを求めることができるのであれば，このアルゴリズムを直接的に適用できる．そうでない場合は，曲面をいったん平面で近似表現すればよい．ただし，平面近似の方法およびその精度などの検討が必要となってくる．

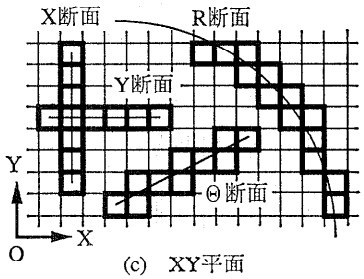
3.2 解析格子の生成

ここで採用する格子自動生成の基本的な考え方は，形状空間の最小分解能の間隔（スキャンラインの間隔）で断面形状の変化を順次追っていき，特徴的な変化のある所に格子面を生成することである．そこで，まず形状の断面図を生成するアルゴリズムについて説明する．

ここで対象としている流体解析では直交座標系と円筒座標系が用いられる．直交座標系の場合は，X, Y, Z 座標軸にそれぞれ垂直な断面を求める必要がある．これらをそれぞれ X 断面，Y 断面，Z 断面と呼ぶこ



(a) 直交座標系 (b) 円筒座標系
(a) Rectangular coord. (b) Cylindrical coord.



(c) XY平面
(c) X-Y plane

図 4 Runlength と断面
Fig. 4 Runlength and section.

とにする（**図4**（a））．また，円筒座標系では，図4（b）に示すように，R, θ, Z 断面を求める必要がある．

Runlength 表現における断面図の生成は，その断面と Runlength を表現する図1に示すような細長い直方体との交差演算となる．しかし，上記の断面はすべてこの特別な場合となり，より単純な演算となる．

Z 断面の場合は，図4（a），（b）に示すように Runlength と断面が垂直となるので，近似的に Runlength をその中心軸で代表させて線分として扱ってよい．断面と Runlength との交点を求め，隣と連続する交点列を接続することにより断面 Runlength を生成する．この場合はピクセルデータのままで処理することも考えられるが，他の断面処理との統一性とデータ圧縮を図るために Runlength 形式とした．

また，Z 断面は他の断面のようにすべての断面図を順次求める必要はない．断面図が変化するのは Runlength の始点および終点においてのみである．そこで，あらかじめすべての Runlength を読み出し，すべての始点値および終点値をチェックしておくことにより処理の効率化を図ることができる．

一方，X, Y, Rおよび θ 断面は Runlength の太さを考慮する必要がある．図4（c）に示すように，Runlength は正方形領域の太さを持つ．そこで，まず断面と交差するこの正方形領域（太線）を求める．X 断面，Y 断面，θ 断面は直線と正方形，R 断面は円と正方形との交差演算となる．そして，この交差する正方形領域上の Runlength が断面 Runlength となる．

この断面図から断面形状の特徴量を求める．本システムではこの特徴量として，**断面変化率** D と**境界線の数** N を採用した．断面図 p と断面図 q との断面変化率 $D(p, q)$ は，次の式で定義される．

$$D(p, q) \equiv S(p \uplus q)/S(p \cup q) \qquad (1)$$

ここで，$\cup$，$\cap$，$\uplus$は，それぞれ形状和，形状積，排他形状和の演算子であり，$S(p)$ は断面図 p の面積を表す．ところで，

$$S(p \cup q) = S(p) + S(q) - S(p \cap q) \qquad (2)$$

$$S(p \cap q) = S(p \cup q) - S(p \uplus q) \qquad (3)$$

であるので，（1）式は，

$$D(p, q) = 2\,S(p \uplus q)/(S(p) + S(q) + S(p \uplus q)) \qquad (4)$$

となる．したがって，断面変化率Dを求めるには，二つの断面図の排他形状和を求めればよい．ただし，R 断面だけは円周方向の長さが異なるので，この方向の長さを正規化して求めている．

排他形状和は，両断面図を表現している Runlength を順次比較し，**図5**に示すように 13 通りの位置関係に対応した Runlength の演算を行うことにより求めることができる．そして，結果として生成された Runlength の長さの総和をとることにより，断面積 S を容易に求めることができる．さらに，本システムでは，より小さな変化も抽出するために，排他形状和をとった断面図における閉領域単位の断面変化率 $d_i (i = 0, 1, 2, \cdots)$ も求めている．

次に，閉鎖領域単位の断面積を求めるアルゴリズムをC言語的記述で**図6**に示す．ここでは，断面図を XY 座標系とし，Runlength の方向を Y 座標軸とする

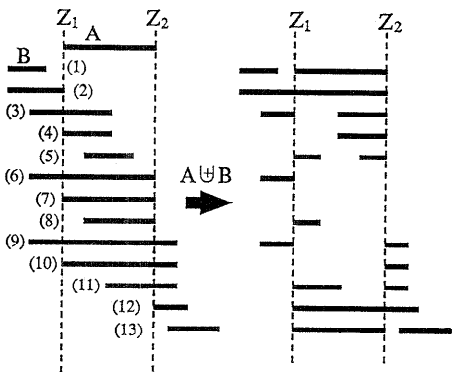


図 5　Runlength の排他形状和
Fig. 5　Exclusive OR of Runlength.

```
n2 = SetRun1( x, v1, v2, v3 );

for ( i = 0, j = 0; i < n1; i++ ) {
  for ( ; j < n2; j++ ) {
    if ( v2[j] < u1[i] ) {
      if ( v3[j] == NULL ) {
        v3[j] = n; s[n] = v2[j]-v1[j]+1; n++; }} —(1)
    else if ( v1[j] < u2[i] ) {
      if ( v3[j] == NULL ) {
        v3[j] = u3[i]; s[u3[i]] += v2[j]-v1[j]+1; } —(2)
      else if ( v3[j] != u3[i] ) {
        u3[i] = v3[j]; s[v3[j]] += s[u3[i]];
        s[u3[i]] = 0.0; }} ——————(3)
    if ( u2[i] < v2[j] ) break; }}

for ( ; j < n2; j++ ) {
  if ( v3[j] == NULL ) {
    v3[j] = n; s[n] = v2[j]-v1[j]+1; n++; }} ——(1)

n1 = n2;
for ( i = 0; i < n2; i++ ) {
  u1[i] = v1[i]; u2[i] = v2[i]; u3[i] = v3[i]; }
```

図 6　面積計算アルゴリズム
Fig. 6　Algorithm of area calculation.

(図 7). x−1 における Runlength を (u1[i], u2[i]) (i=0, 1, ⋯, n1−1), この Runlength が属する領域の番号を u3[i] とする. 同様に, x における Runlength を (v1[j], v2[j]) (j=0, 1, ⋯, n2−1), その領域番号を v3[j] とする.

閉領域番号 n, 閉領域番号 n の面積 s[n] (n=0, 1, 2, ⋯) および Runlength の数 n1 をすべてゼロと初期設定して, すべての x に関して図 6 に示す処理を行うことにより, s[n] に閉領域番号 n の断面積が求まる.

図 6 における SetRun 1 ルーチンは, x におけるすべての Runlength を (v1, v2) にセットし, すべての v3 を NULL (「空」を意味する) に初期化する. そして, その Runlength の数を返す. 図 6 の処理(1)は, 図 7 の(1)に示すように, Runlength (v1[j], v2[j]) が (u1[i−1], u2[i−1]) および (u1[i], u2[i]) のいずれとも隣接せず, その間となる場合である. したがって, (v1[j], v2[j]) は新しい閉領域を形成する可能性があるので, v3[j] には新しい閉領域番号 n を代入し, s[n] には Runlength の長さ v2[j]−v1[j]+1 を代入する. 同様に, 処理(2)は, (v1[j], v2[j]) が (u1[i−1], u2[i−1]) とは隣接せず, (u1[i], u2[i]) と接する場合である. 処理(3)は, (v1[j], v2[j]) が (u1[i−1], u2[i−1]) および (u1[i], u2[i]) と接し, かつ領域番号 u3[i−1] と u3[i] が異なる場合である. この場合は両者の領域は同一となるので統合する.

以上から, d_i および D は次の式により求めることができる.

$$d_i(p,q)=2s[i]/(S(p)+S(q)+\sum_i s[i]) \quad (5)$$

$$D(p,q)=\sum_i d_i(p,q) \quad (6)$$

次に, 境界線の数 N を求めるアルゴリズムについて説明する. 境界線自体を求め, それらを数え上げる方法もあるが, ここでは, 境界線を求めることなく, 直接的に境界線の数を数え上げるアルゴリズムを提案する.

閉領域の面積計算の場合と同様に, x−1 における Runlength を (u1[i], u2[i]) (i=0, 1, ⋯, n1−1), この Runlength の始点側の境界線番号を u3[i], 終点側の境界線番号を

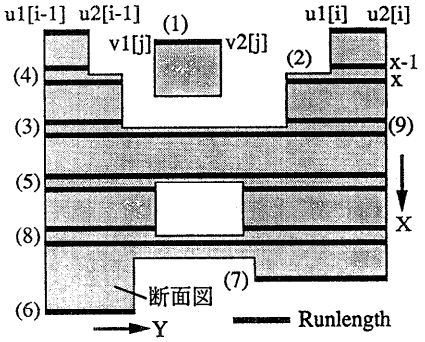


図 7 断面図における Runlength のパターン
Fig. 7 Runlength classifications of a section.

```
n2 = SetRun2( x, v1, v2, v5 );

for ( i = 0, j = 0; i < n1; i++ ) {
  for ( ; j < n2; j++ ) {
    if ( v2[j] < u1[i] ) {
      if ( v5[j] == OFF ) { v3[j] = v4[j] = n; n++; } ---------(1)
      else                { v4[j] = u4[i-1]; }} ---------(4)
    else if ( v1[j] <= u2[i] ) {
      if ( u5[i] == OFF ) {
        if ( v5[j] == OFF ) {
          v3[j] = u3[i]; v4[j] = u4[i]; u5[i] = v5[j] = ON; } (2)
        else {
          v4[j] = u4[i]; u5[i] = LINK; }} ---------(3)
      else if ( v5[j] == OFF ) {
        v4[j-1] = v3[j] = n; n++; v4[j] = u4[i];
        v5[j] = LINK; }}                              (5)
    if ( u2[i] < v2[j] ) break; }}

for ( ; j < n2; j++ ) {
  if ( v5[j] == OFF ) { v3[j] = v4[j] = n; n++; }} ---------(1)

for ( i = 0; i < n1; i++ ) {
  j = NULL;
  if ( u5[i] == OFF ) {
    if ( u3[i] == u4[i] ) { N++; } ---------(6)
    else { j = u3[i]; k = u4[i]; }} ---------(7)
  else if ( u5[i] == LINK ) {
    if ( u4[i-1] == u3[i] ) { N++; } ---------(8)
    else { j = u4[i-1]; k = u3[i]; }} ---------(9)
  if ( j == NULL ) continue;
  for ( h = 0; h < n1; h++ ) {
    if ( u3[h] == k ) u3[h] = j;
    if ( u4[h] == k ) u4[h] = j; }
  for ( h = 0; h < n2; h++ ) {                        (10)
    if ( v3[h] == k ) v3[h] = j;
    if ( v4[h] == k ) v4[h] = j; }}

n1 = n2;
for ( i = 0; i < n2; i++ ) {
  u1[i] = v1[i]; u2[i] = v2[i]; u3[i] = v3[i];
  u4[i] = v4[i]; u5[i] = OFF; }
```

図 8 境界線数算出アルゴリズム
Fig. 8 Algorithm of boundary number calculation.

u4[i]，その状態変数を u5[i] とする．同様に，xにおける Runlength を (v1[j], v2([j]) (j=0, 1, …, n2−1)，その両端における境界線番号をそれぞれ v3[j], v4[j]，状態変数を v5[j] とする．

境界線の数 N，境界線番号 n および Runlength の数 n1 をすべてゼロと初期設定し，**図8**に示す処理をすべてのxに関して行うことにより，境界線の数 N を求めることができる．ここで，ON は隣合う Runlength が接していることを，OFF は接していないことを表し，LINK は Runlength が同じスキャンライン上の一つ前の Runlength と同じ境界線を共有することを表す．

図8における SetRun 2 ルーチンは，xにおけるすべての Runlength を (v1, v2) にセットし，すべての v5 を OFF に初期化する．そして，その Runlength の数を返す．図8の処理(1)は，図7の(1)に示すように，Runlength (v1[j], v2[j]) が (u1[i−1], u2[i−1]) および (u1[i], u2[i]) のいずれとも隣接せず，その間となる場合である．(v1[j], v2[j]) は新しい境界線を形成する可能性があるので，v3[j], v4[j] には新しい境界線番号 n を代入する．同様に，処理(4)は，(v1[j], v2[j]) が (u1[i−1], u2[i−1]) と接する場合である．v3[j] はそれまでの処理で求まっている．処理(2)は，(v1[j], v2[j]) が (u1[i], u2[i]) と接する場合である．処理(3)は，(v1[j], v2[j]) が (u1[i−1], u2[i−1]) および (u1[i], u2[i]) と接する場合である．すなわち，(u1[i−1], u2[i−1]) と (u1[i], u2[i]) が (v1[j], v2[j]) を介して接続する．処理(5)は，(v1[j], v2[j]) が (u1[i], u2[i]) と接し，かつ (u1[i], u2[i]) を介して (v1[j−1], v2[j−1]) と接続する場合である．そこで，新しい境界線を形成する可能性がある．

処理(6)〜(9)は，Runlength (u1[i], u2[i]) において，境界線が閉じたかどうかを判定する処理である．すなわち，処理(6)では，図7の(6)に示すように，隣接する Runlength (v1[j], v2[j]) が存在しないので，Runlength の始点側と終点側の境界線がつながる．そして，u3[i] と u4[i] が等しいのでこの境界線は閉じたことになる．処理(7)では，異なる境界線番号 u3[i] と u4[i] を持つ境界線がつながるので，境界線番号が u4[i] であるものすべてを u3[i] とする (処理(10))．同様に，処理(8)では境界線が閉じたことになり，処理(9)では，境界線番号 u4[i−1] と u3[i] の境界線はつながる．

本システムでは，この二つの断面特徴量を用いて，以下の判別式により格子面を自動生成している．すなわち，次の3式のうち少なくとも一つが成立した場合に，断面 p は格子面となる．ここで，d_{max} は d_i の最大値を，$N(p)$ は断面 p における境界線の数を表し，α, β は適当なしきい値である．

$$D(p-1, p)-D(p-2, p-1)-D(p, p+1)>\alpha \quad (7)$$

$$d_{max}(p-1, p)-d_{max}(p-2, p-1)-d_{max}(p, p+1)>\beta \quad (8)$$

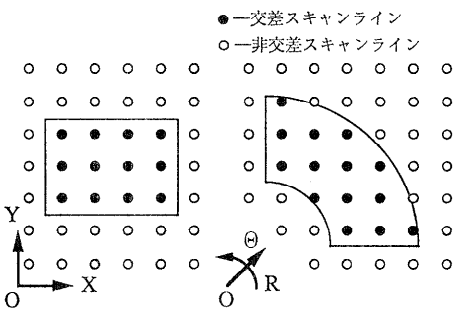
$$(N(p-2)=N(p-1))\wedge(N(p)=N(p+1))\wedge(N(p-1)\neq N(p)) \quad (9)$$

また，本システムではこのようにして求めた格子データを対話的に修正することができる．解析の計算精度を確保するためには隣合う格子面の間隔の比は1.5倍以下であることなど，解析プログラム側の制約条件がある．上記したような断面特徴量のみを用いた格子生成では，このような形状にかかわらない条件まで加味することはできない．そこで，格子データの対話修正が必要となってくる．

3.3 流体占有率・開口率の計算

格子面により生成される小立体はセルと呼ばれ，直交座標系では，セルは直方体（図4(a)），円筒座標系では扇柱（図4(b)）となる．そして，流体占有率の計算はこのセルに含まれる形状の体積を，開口率の計算はセル面に形状が占める断面積を求めればよい．セル面上に形状が占める断面積は，格子生成における断面積計算で述べたのと同じ方法により求めることができる．

セル内部に占める形状の体積を求めるには，まず，セルの底面と交差するスキャンラインを求める．この



(a) 直方体セル (b) 円筒セル
(a) Rectangular cell (b) Cylindrical cell

図9 Runlength とセル
Fig. 9 Runlength and cell.

場合はスキャンラインの太さを考慮する必要はなく，近似的に線分として扱ってよい．すなわち，直方体セルの場合はスキャンラインと長方形（**図 9**（a）），円筒セルの場合はスキャンラインと扇形（図9（b））との内外判定を行う．次に，これらのセルと交差するスキャンライン上の Runlength でセル内部に含まれる部分の長さを求めその総和をとる．

セルと交差するスキャンラインの数が十分でない場合，すなわちセルの体積が小さく，セルに対する分解能が低い場合は，流体占有率・開口率の計算精度が低下する．そこで，本システムでは特定の（体積の小さい）セルに対して，分解能を上げて（スキャンラインの数を増やして），そのセルと交差するスキャンラインを求め直し，そのスキャンライン上の Runlength データのみを原データ（CSG，BRep）から再生成することができる．これにより，形状空間全体の分解能を上げる場合に比べて，少ない記憶領域と演算量で特定のセルの流体占有率・開口率が再計算でき，その精度を改善することができる．

4. 評価と考察

実際の解析事例における本システムの評価結果を報告する．直交座標系の解析事例として半導体製造装置（Chemical Vapor Deposition）および自動車の室内，円筒座標系の事例として磁気ハードディスク装置（HDD）を取り上げた．

その結果は**表 1** となった．処理時間はすべて CPU 時間である．Runlength 表現は座標系に対してユニークではないので，Runlength 生成に関してはスキャンラインを X，Y，Z 座標軸および斜視図方向に取り，その中での最小値（左欄）および最大値（右欄）を測定した．形状空間の分解能は 1024^3 とし，形状は形状空間内にほぼ一杯になるように生成した．CVD および HDD は BRep モデルから，自動車は CSG モデルから生成した．なお，BRep は三角形面のみで構成されていた．計算機は HP 9000/750 CRX 24（76 MIPS，64 MB）を使用し，プログラムはすべてC言語で書き，表示は X-window を用いた．

これらの事例の出力例を**図 10**（a）～（l）に示しておく．これらの表示画像はいずれも，Runlength 表現から生成されている．なお，図 10（g），（h）の格子図は自動生成のみによるものであり，他は自動生成後対話的に修正されている．

（1） アルゴリズムの処理時間について

CSG，BRep→Runlength 変換，格子生成および流体占有率・開口率計算は，それほどリアルタイム性は要求されないので，最悪値で数分のオーダであれば十分であろう．

（2） データ圧縮性について

本システムの Runlength データの属性データは 4 バイト（形状番号 2，色番号 2）で構成されている．同様のことを Voxel で実現すると，$1024^3 \times 4 = 4$ ギガバイトのメモリが必要である．さらに，2048^3 以上の高分解能化などを考えると，非常に多くのメモリを必要とする．

一方，Runlength 表現では，今回の事例では高々十数メガバイトのオーダで済み，2048^3 あるいは 4096^3 レベルの高分解能にもメガバイトのオーダで済むことが予測される．これは，今回取り上げた流体解析分野においては，その対象となる形状にはそれほど凹凸がないことによる．このような対象に対しては Runlength 表現が非常に有効であるといえる．

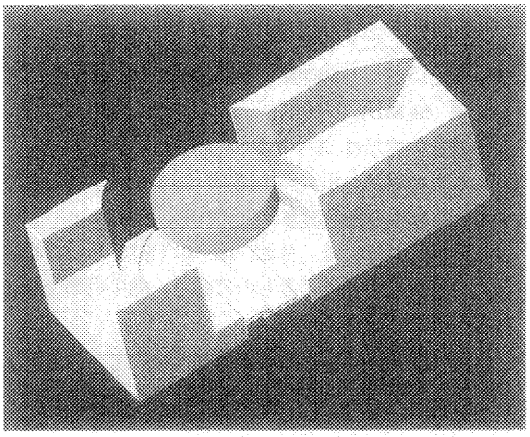
（3） 座標系に対するユニーク性の欠如について

今回の結果から判断する限りでは，処理時間，データ量共に最悪値で 3 倍程度増えている．この程度であれば Runlength 表現のユニーク性の欠如は実用上そ

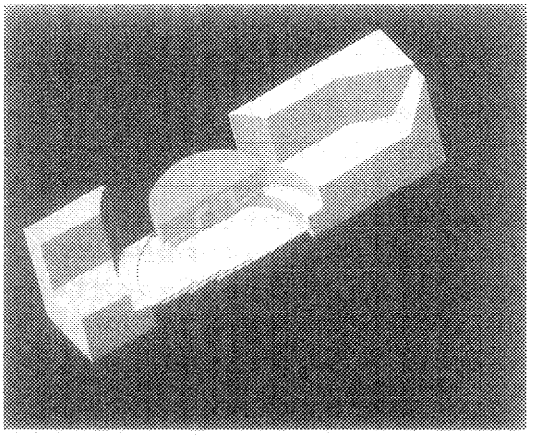
表 1 事例の評価結果
Table 1 Processing time and data amount of the examples.

		CVD		自動車		HDD	
Runlength生成	面の数／プリミティブ数	1,664 faces		110 primitives		2,749 faces	
	変換時間（秒）	2.16	4.56	8.09	10.90	2.63	8.31
	データ量 (MBytes)	6.65	11.05	7.50	9.94	7.52	17.34
自動格子生成	格子数 (X×Y×Z, R×Θ×Z)	5,040(14×18×20)		41,492(44×41×23)		4,940(19×20×13)	
	生成時間（秒）	176.9		173.4		148.9	
流体占有率／開口率計算	格子数 (X×Y×Z, R×Θ×Z)	4,485(13×15×23)		318,500(70×70×65)		6,688(19×22×16)	
	計算時間（秒）	2.49		48.26		17.66	

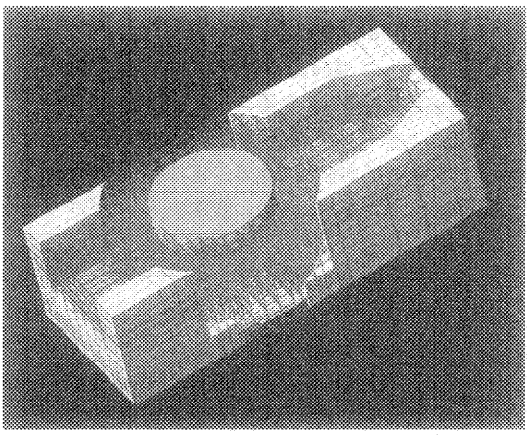
□ — 設定値　□ — 出力値・測定値　（形状空間分解能:1024×1024×1024）



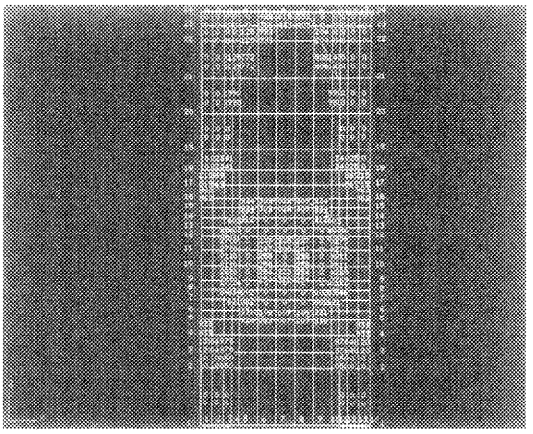
(a) CVD 全体図
(a) Perspective view of the CVD.



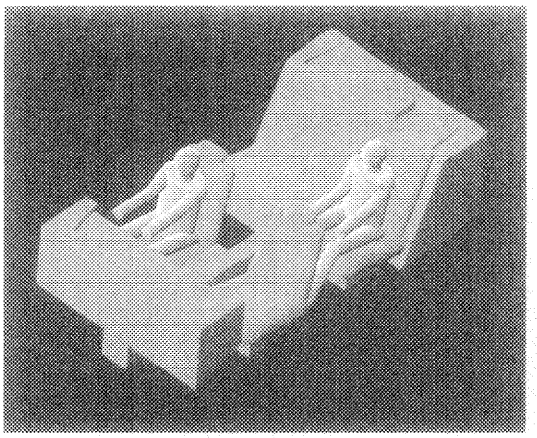
(b) CVD の断面図
(b) Cross-sectional view of the CVD.



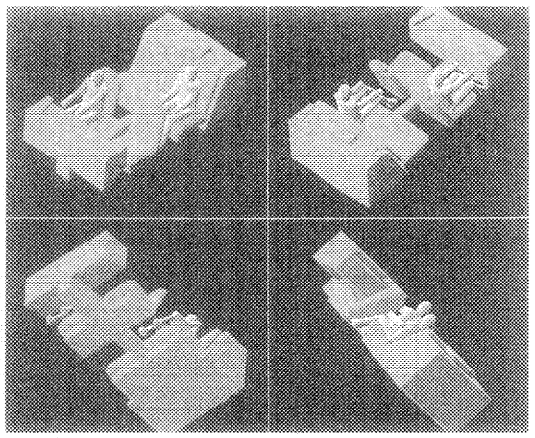
(c) CVD の格子図
(c) Grid view of the CVD.



(d) CVD の流体占有率・開口率表示
(d) Porosity display of the HDD.

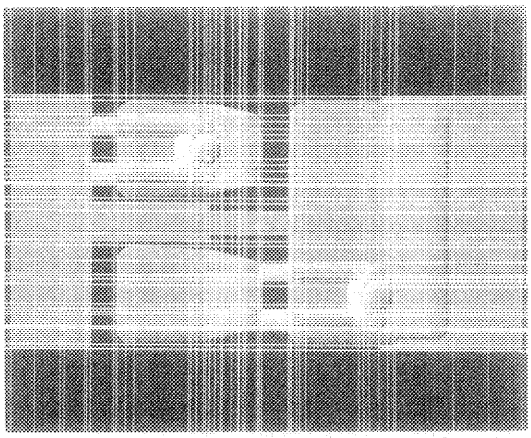


(e) 自動車室内全体図
(e) Perspective view of the cabin.

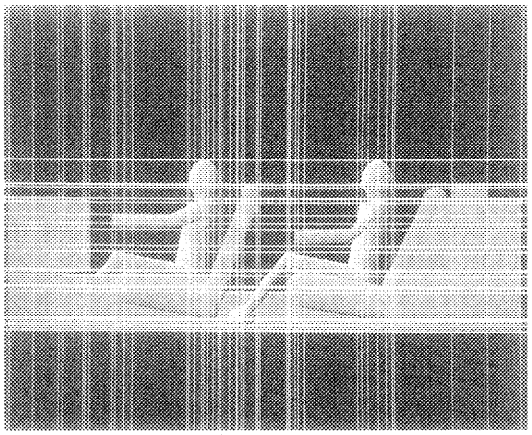


(f) 自動車室内のマルチビュー
(f) Multi view of the cabin.

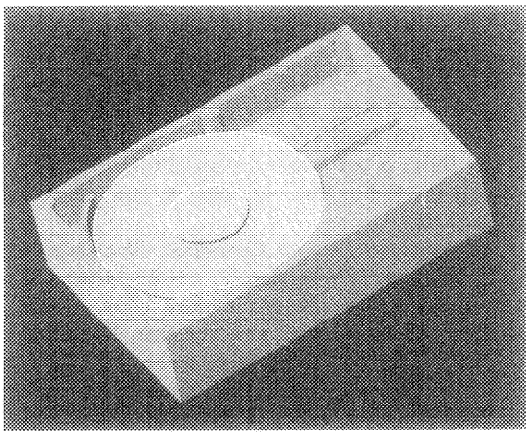
図 10 グラフィック出力例
Fig. 10 Examples of the graphic output.



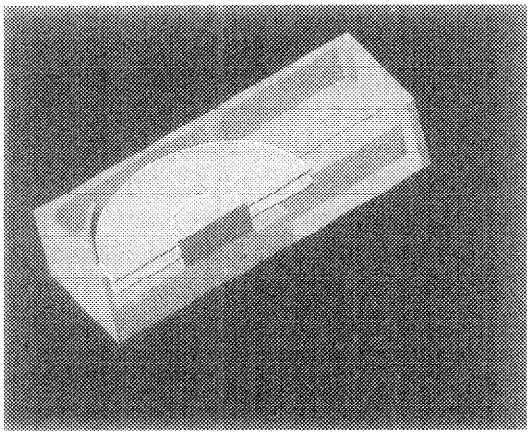
(g) 車内の格子図 (*X-Y* 平面)
(g) Grid display of the cabin (*X-Y* section).



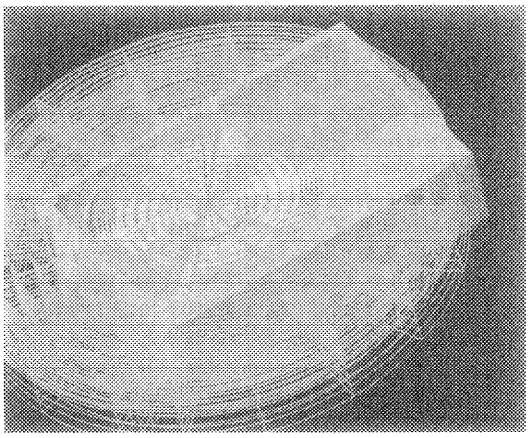
(h) 車内の格子図 (*X-Z* 平面)
(h) Grid display of the cabin (*X-Z* section).



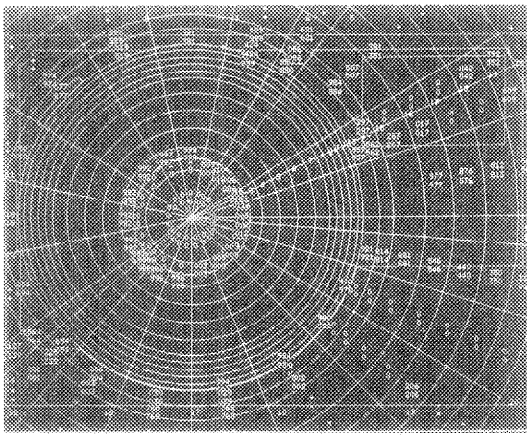
(i) HDD 全体図
(i) Perspective view of the HDD.



(j) HDD の断面図
(j) Cross-sectional view of the HDD.



(k) HDD の格子図
(k) Grid view of the HDD.



(l) HDD の流体占有率・開口率表示
(l) Porosity display of the HDD.

図 10 (つづき)
Fig. 10 (Continued)

れほど問題にならないと判断される．

（4） 属性データについて

CVD は，図 10（a）に示すように形状は黄色と青色の部分から構成されているが，両者は同一の形状番号を持つ．すなわち同じ物質定数を持つとして解析される．一方，HDD はディスク部（ベージュ色）とシャーシ部（青色）では異なる形状番号を持ち（図 10（i）），異なる物質定数を持つとして解析される．このように，本システムでは属性データを用いて「複数固体解析」に対応している．

（5） 隠線・隠面処理について

図 10（c）および（k）では，形状と格子データ（直線，円弧）が混在して隠線・隠面処理されている．このような表示画像の生成においては，まず，これらの体積を持たない線分や円弧も近似的に Runlength 表現される．すなわち，線の場合は最小太さ（スキャンラインの間隔），面の場合は最小厚みを持つ 3 次元立体として処理される．そして，形状を表現している Runlength と一緒にされて，表示の視線方向から見た Runlength の前後判定が行われる．すなわち，一番前面にある Runlength のみが取り出され，表示画像が生成される．このように，Runlength 表現では，非立体である線，面も Octree などと比べてそれほど問題なく近似表現でき，これらの非立体が混在した隠線・隠面処理が Runlength の演算で統一的に行うことができる．ただし，この方式はエイリアシングが生じやすいという欠点を持つ．

5. おわりに

Runlength 形式を用いて流体解析前処理システムを構築し，次の三つの機能を実現した．

（1） BRep/Runlength 変換

（2） 解析格子の自動生成

（3） 流体占有率・開口率の自動計算

そして，その有効性・実用性を実際の事例を用いて実証した．

今後，差分法を用いた流体解析においては，計算機能力の増大にともなって，解析精度をより向上させるために解析格子はより細かくなり，並列処理化も進展するであろう．そして，対象形状はより複雑化し，より一層の高分解能化も必要となってくるであろう．

したがって今後の課題としては，このような形状の複雑化，高分解能化および解析格子の増大に対して，今回提案したアルゴリズムの演算量がどう変化するかを捉え，並列処理化の検討をすることである．

もう一つの課題としては，解析プログラム側の制約条件を含んだ解析格子自動生成がある．今回提案した断面変化による自動生成は形状情報のみを用いたものであり，解析のために最適な格子とするには，格子を修正しなければならない場合も多い．したがって，この両者をうまく融合することが必要である．

さらに，有限要素法，Boundary fit 法などの今回は触れなかった他の流体解析手法に対する Runlength 形式の適用性を探ることも意義のあることであろう．

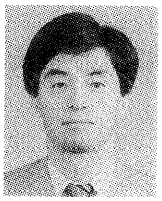
謝辞 本研究をまとめるにあたり貴重な御助言，御協力を頂いた，松下電器産業(株)の古山氏，および通信総合研究所知覚機構研究室の野田室長に厚くお礼を申し上げる．

参考文献

1) 中橋ほか：小特集「グリッド・ジェネレーションとその応用」，情報処理，Vol. 30, No. 7, pp. 766-788 (1989).
2) 栗山：エンジンルーム内の三次元熱流体解析，自動車技術，Vol. 42, No. 8, pp. 1083-1091 (1988).
3) Baumgart, B. G.: *Geometric Modeling for Computer Vision*, Rep. STAN-CS-74-463, Computer Science Dept., Stanford Univ. (1974).
4) Requicha, A. A. G. and Voelcker, H. B.: Solid Modeling: A Historical Summary and Contemporary Assessment, *IEEE Computer Graphics & Applications*, Vol. 2, No. 2, pp. 9-24 (1982).
5) Requicha, A. A. G. and Voelcker, H. B.: Solid Modeling: Current Status and Research Directions, *IEEE Computer Graphics & Applications*, Vol. 3, No. 7, pp. 25-37 (1983).
6) Jackins, C. L. and Tanimoto, S. L.: Oct-Trees and Their Use in Representing Three-Dimensional Objects, *Computer Graphics and Image Processing*, Vol. 14, No. 3, pp. 249-270 (1980).
7) Meagher, D.: Geometric Modeling Using Octree Encoding, *Computer Graphics and Image Processing*, Vol. 19, No. 2, pp. 129-147 (1982).
8) 登尾，福田，有本：BRep からオクトツリーへの変換アルゴリズムとその評価，情報処理学会論文誌，Vol. 28, No. 10, pp. 1003-1012 (1987).
9) Kaufman, A. and Bakalash, R.: Memory and Processing Architecture for 3D Voxel-Based Imagery, *IEEE Computer Graphics & Applications*, Vol. 8, No. 6, pp. 10-23 (1988).
10) Atherton, P. R.: A Method of Interactive Visualization of CAD Surface Models on a Color Video Display, *Computer Graphics*, Vol. 15, No. 3, pp. 279-287 (1981).

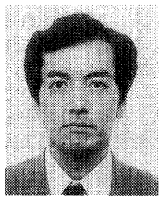
11) Martin, W. N. and Aggarwal, J. K.: Volumetric Descriptions of Objects from Multiple Views, *IEEE Trans. Patt. Anal. Mach. Intell.*, Vol. PAMI-5, No. 2, pp. 150-158 (1983).
12) Okino, N., Kakazu, Y. and Morimoto, M.: Extended Depth-Buffer Algorithms for Hidden-Surface Visualization, *IEEE Computer Graphics & Applications*, Vol. 4, No. 5, pp. 79-88 (1984).
13) Hook, T. V.: Real-time Shaded NC Milling Display, *SIGGRAPH '86*, Vol. 20, No. 4, pp. 15-20 (1986).
14) 川島，太田，徳増：CAD 対話インタフェースのためのソリッドモデル隠面処理法，情報処理学会論文誌，Vol. 28, No. 2, pp. 147-155 (1987).
15) Roth, S. D.: Ray Casting for Modeling Solids, *Computer Graphics and Image Processing*, Vol. 18, No. 2, pp. 109-144 (1982).
16) Boyse, J. W. and Gilchrist, J. E.: GMSolid: Interactive Modeling for Design and Analysis of Solids, *IEEE Computer Graphics & Applications*, Vol. 2, No. 2, pp. 27-40 (1982).
17) Lee, Y. T. and Requicha, A. A. G.: Algorithms for Computing the Volume and Other Integral Properties of Solids, *Communications of the ACM*, Vol. 25, No. 9, pp. 635-650 (1982).
18) 荒川，三谷，一柳：3D ランレングス法ソリッドモデラによる流体解析プリプロセッサ，第1回数値流体力学シンポジウム講演論文集，pp. 351-354 (1987).
19) 荒川：仮想空間における立体形状モデリング，'91 グラフィクスと CAD シンポジウム論文集，Vol. 91, No. 7, pp. 33-42 (1991).

(平成5年1月26日受付)
(平成5年6月17日採録)



荒川　佳樹（正会員）

1954年生．1978年早稲田大学理工学部工業経営学科卒業．1980年同大学院理工学研究科修士課程修了．同年松下電器産業(株)入社．1990年郵政省通信総合研究所入所．現在，同研究所関西先端研究センター知覚機構研究室に勤務．形状モデリングおよび形状認識・理解の研究に従事．PCSJ '93 実行委員．電子情報通信学会，精密工学会各会員．



三谷　眞人

1958年生．1981年東京工業大学工学部生産機械卒業．1983年同大学院修士課程精密機械システム修了．同年松下電器産業(株)入社．以来，半導体製造装置内のガス流動数値解析，磁気テープ高速塗工工法の研究開発，差分流体解析ソルバー・前処理ソフトの開発，差分流体解析用並列計算アルゴリズムの研究開発に従事．日本機械学会，日本流体学会各会員．

一柳　高時

1945年生生．1968年東京工業大学工学部機械卒業．1970年同大学院修士課程修了．同年松下電器産業(株)入社．以来，構造解析・熱流体解析・電磁気解析などシミュレーションを利用したホームエレクトロニクス製品開発と CAD/CAM/CAE の研究開発に従事．現在，同社生産技術研究所 CAE 研究部部長．日本機械学会会員．