スペースモデルによる二次元図形処理システムの試作†

——注目，局所集合演算，運動図形の衝突検出——

大沢 晃†† 川崎敏治††† 岩本哲夫†††

スペース・モデリング[1),2)]は図形空間をポインタで表現し，局所処理により図形処理を効率化する，二次元・三次元一貫した新しい図形データ構造のコンセプトである．スペースモデルの応用範囲は広いが，今回の試作は基本原理・性能の確認を目的とし，二次元多角形に関するデータ構造生成，注目，隣接空間検索，図形集合演算，運動図形の衝突検出，表示に範囲を限定した．実験結果は原理どおり，局所処理により良好な対話性能を得ることができた．必要な図形への注目時間は CPU 時間で 4〜5 ms (CPU 68020，演算コプロセッサ付き)，隣接空間検索は 200 μs，注目した多角形と他の多角形との集合演算は 1 ms 以下，衝突検出は運動経路長に比例するが短距離ならば 1 秒以下と高速である．これらは原理的に全体図形数によらず $O(1)$ の処理である．また大量図形の一括データ構造生成時間は一頂点（交点を含む）あたり 4 ms 以下となり，これも原理どおり全体で $O(N)$ (N は交点を含む総頂点数) であることがデータで裏づけられた．その他，表示すべき重なり図形を後から塗消さないため，本モデルの包含図形検索機能を使って，常に外側図形から表示するよう高速に表示順序を決める方式もテストした．

1. まえがき

LSI，地図，等大規模図形を取り扱う図形処理システムでは，従来から集合演算を含む図形処理の高速化手法が研究されてきた[5)-7)]．しかし，これらの多くは図形全体のバッチ処理に関するものであり，対話処理に関する研究が待たれていた[8)]．対話処理では目的とする部分のみに注目した局所的処理によって応答を速くすることが必要になる．この目的で Ousterhout らは「Corner Stitching」と呼ぶデータ構造を提案し LSI 用対話形 CAD に適用した[3),4)]．しかしこの手法は二次元で，かつ図形形状が垂直・水平方向の辺よりなる矩形の組合せに限定される問題があった．これに対しスペースモデル[1),2)]は部分空間をポインタで表現することにより，そのような制限なしで局所的図形処理を可能とする，二次元・三次元一貫した図形データ構造である．文献 1) と重複するが，説明の都合上第2章で二次元モデルの基本原理を概説する．スペースモデルではデータ構造を一度作成しておけば，隣接・重なり図形の検索が，図形空間の一部に注目した局所的処理で可能になる．このため，局所的図形集合演算，運動図形の衝突検出，近傍の探索，等が極端な特殊形状の場合を除き $O(1)$ (総図形頂点数 N に依存しない) の時間で実現する．本論文はその基本原理と基本性能の確認を目的として行った二次元原理試作に関する報告である．今回は実現してないが最終的な用途としては，二次元では LSI 用対話形 CAD，地図，三次元では機械系 CAD/CAM，隠面・隠線消去，ロボットの運動シミュレーション，等が考えられる．

第3章では試作の概要について述べる．対象図形は簡単のため多角形のみとし，処理内容もデータ構造生成，局所集合演算，運動図形の衝突検出，結果の表示のみに絞った．第4章では性能評価について述べる．試作の結果，一部への注目処理，隣接空間検索，局所的集合演算，運動図形の衝突検出，が全体図形数にかかわらず対話速度で実行できることが明らかになった．なお，重なり図形の表示では，表示すべき図形を後から塗消さないための，表示順序決定手法を開発した．

その他基本原理を具体化するには，各種特殊ケース対策を行う必要がある．第5章で対策の一端を述べる．

2. 二次元スペースモデルの原理

2.1 基本原理

ここでは多角形で説明する．**図1**のごとく，頂点の周りの凹な（辺に挟まれた角度が 180° より大な）空間を，頂点を通る Y 軸に平行な仮想線 B_x (図中一点鎖点) で区切る[*]．そうすると B_x と各図形の辺 B_y によって全体が部分空間 $\omega_0, \cdots, \omega_i, \cdots$ に分割される．こ

† Prototyping of 2D Geometrical Process Using Space Model: Focussing, Local Set Operation, and Collision Detection by AKIRA OHSAWA (Hitachi Keihin Technical College), TOSHIHARU KAWASAKI and TETSUO IWAMOTO (Microelectronics Products Development Lab., Hitachi Ltd.).

†† 日立京浜工業専門学院電子工学科
††† (株)日立マイクロエレクトロニクス機器開発研究所

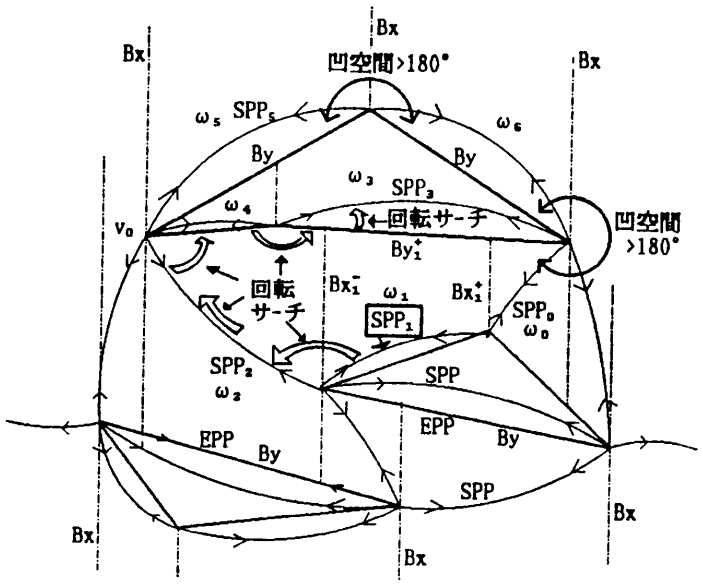


図 1 部分空間 ω_i の SPP_i による表現と隣接空間の検索
Fig. 1 Representation of subspace ω_i by SPP_i, and retrieval of neighborhood.

の場合 B_x が B_y を突き抜けることはないとする．空間を頂点を通る B_x と B_y とで区切ったのであるから，各 ω_i はその X^+, X^- 端（X 座標値最大および最小の位置）に必ず頂点をもつ（辺と辺の交点も頂点とする）．その頂点にそれぞれの空間**に対応するポインタ SP (Space Pointer) を設置し，ω_i の両端の SP が相互に相手の SP のアドレスを指すようにする．この SP の対を SPP (Space Pointer Pair：図中矢印付細線）と呼ぶ．SPP を設置した全体の図形データ・ファイルを SPF (Space Pointer File) と名付ける．

X 座標値の等しい 2 個以上の頂点が存在するときには，それらが X 方向に相互にごくわずかずれていると考えれば，どの ω_i の左右端にもそれぞれ各 1 個で，かつ 1 個に限る頂点が存在することになる．これにより，各 ω_i には必ず 1 組の SPP_i が 1 対 1 で対応することになる．1 対 1 なので，SPP_i が与えられたとき対応する ω_i の持つ情報を求めることが容易なら，SPP_i は ω_i を表現していると言ってよい．そうすれば，図形処理における図形（空間）検索問題は SPF の中から対応する SPP_i を検索する問題に置換できる．ここで ω_i の持つ情報とは，ω_i の外形線となる仮想線 B_x と図形辺 B_y，これらに接する X, Y 側隣接空間，色，図形名等である．

例えば図 1 で SPP_1 が与えられたとき，ω_1 の X^+ 側外形線 Bx_1^+ や，隣接空間を表現する SPP_0 は，ポインタをたどって容易に検索できる．また Y^+ 側外形線 By_1^+ は，SPP_1 から図中白矢印 ↶ で示すようにポインタをたどる回転サーチと名付けた手法により求められる．回転サーチは SPP の接続する頂点（同図 V_0）が極端に遠方にある例外的な図形を除けば局所的であり，全体図形数に無関係な $O(1)$ の処理となる．有限な図形なら V_0 は必ず存在する．Y^+ 側隣接空間を表現する SPP_3 も，By_1^+ からの回転サーチで検索できる．このように X 側，Y 側ともに隣接空間がポインタで検索できるから，隣接空間伝いに任意の場所へ自由に注目点を移動させることができる．

SPF では図形辺 B_y も辺を表現するポインタ EP (Edge Pointer) の対よりなる EPP (Edge pointer Pair) で表現する．また色彩や図形名等は SPF 生成時に**図 2** A∩B のごとく SPP の空間属性として付加する．これにより図形の頂点や交点は EPP または SPP の接続点として，図形の重なりは空間属性の重なりとして認識できる．重なりが分かれば図形集合演算は容易である．例えば図形の AND は重なり部分のみを取り出せばよいし，OR は重なり数にかかわらず図形内部の属性を持つ空間を出力すればよい．

2.2 SPF データ構造の生成法

試作の中心課題について述べる．SPF データ構造の生成は，r 個の入力図形についてのデータ構造 SPF_r が既に生成されているとして，これに第 $r+1$

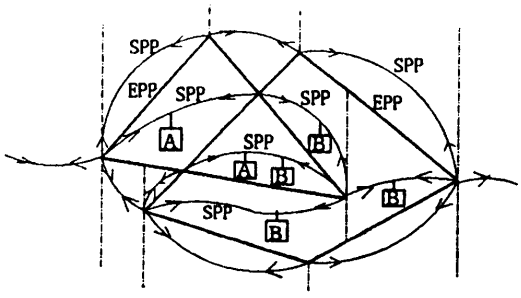


図 2 空間属性 A∩B による重なりの表現
Fig. 2 Representation of overlap by space attributes.

☆ ここでは図の煩雑さを避けるために B_x を凹な空間に限定したが，凸な（180° 以下の）空間にもすべて B_x を設置する方式もある（文献 1，付録 1）．
☆☆ 以下，部分空間のことを単に空間とも呼ぶ．

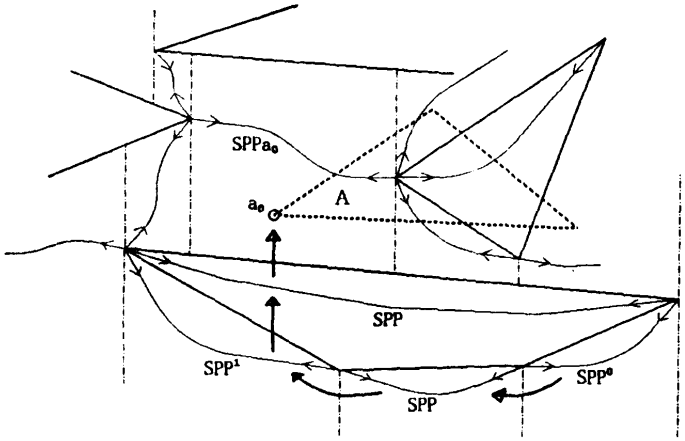


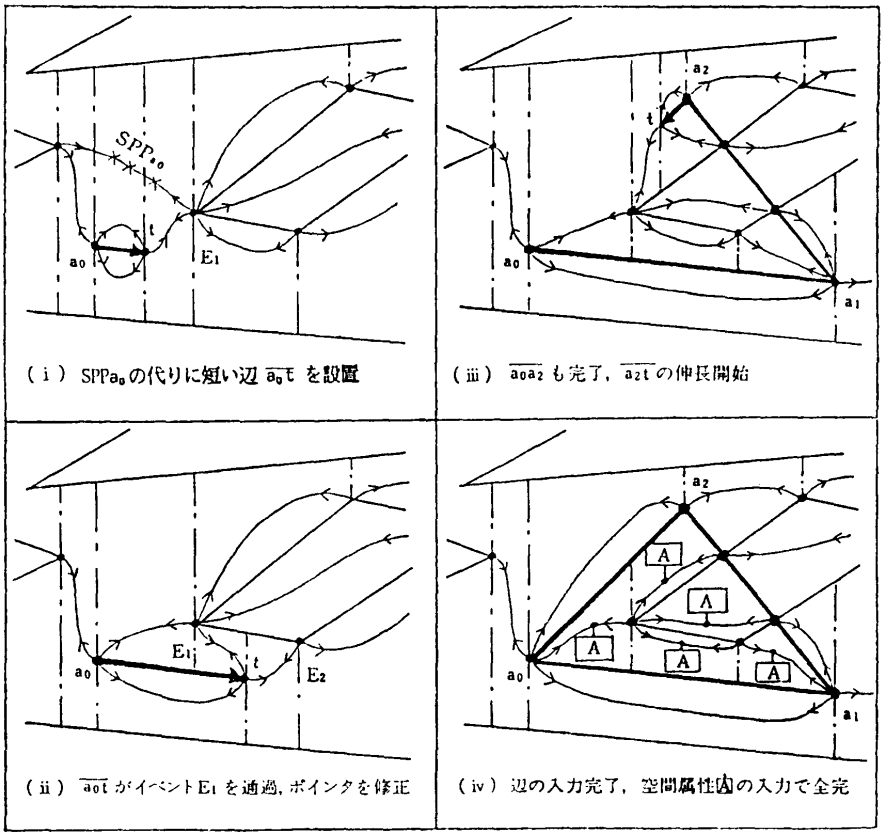
図 3 点 a_0 を含む部分空間 SPP_{a_0} の検索
Fig. 3 Retrieval of subspace SPP_{a_0} including point a_0.

番目の図形データを追加して SPF_{r+1} を作ることの繰返しで行う．以下 SPF_r に頂点列 A $\{a_0(x_0, y_0), a_1(x_1, y_1), \cdots, a_i(x_i, y_1), \cdots\}$ で与えられた多角形 A を追加する方法を説明する．

（1） はじめに図 3 の SPF_r で，A の頂点 a_0 がどの空間（SPP_{a_0} で示す）に位置するかを調べる．SPP_{a_0} を求めるには最初に任意の SPP^0 をとり，そこから隣接空間伝いに点 $a_0(x_0, y_0)$ に接近する手法を取る．その際同図黒矢印のごとく，まず x_0 を含む空間 SPP^1 に達するまで X 方向へ進み，次に Y 方向へ進んで SPP_{a_0} に至る経路をとる．X 方向への接近は頂点の X 座標値を調べながら x_0 の方向へ SPP をたどればよい．Y 方向へは前節の回転サーチによって Y 側の隣接空間を検索しながら進む．SPP_{a_0} への到着は，a_0 が SPP_{a_0} の空間外形の B_y より内側に在ることを計算して判定する．

（2） SPP_{a_0} へ到着したら図 4（i）のごとく短い辺 a_0t を挿入する．

（3） 次に図 4（ii）～（iii）のごとく a_0t から始めて A の辺に沿って折線 a_it（i は頂点番号）を伸ばし，



（i） $SPPa_0$ の代りに短い辺 $\overline{a_0t}$ を設置
（iii） $\overline{a_0a_2}$ も完了，$\overline{a_2t}$ の伸長開始
（ii） $\overline{a_0t}$ がイベント E_1 を通過，ポインタを修正
（iv） 辺の入力完了，空間属性Aの入力で全完

図 4 図形 A の入力処理
Fig. 4 Input process of figure A.

空間分割の変化に応じて SPP の設置，更新を行う．空間分割の変化は $a_i t$ の先端が既存の B_x または B_y を貫通する時点，または A の頂点に達したとき発生する．$a_i t$ が A の辺を一巡すれば辺の入力は終わる．

(4) 最後に図4(iv)のごとく A の内部の SPP に空間属性を付加する．

以上のうち(2)～(4)は A の周囲空間に限定された局所的処理であるから，特殊な場合（文献1付録）をのぞけば $O(1)$ の処理になり高速と言える．

これに対し(1)の処理は $O(\sqrt{N})$ となる（N 個の頂点・交点を含む二次元空間の中を SPP^0 から SPP_{a_0} まで一次元的に進む距離）ので対策を要する．**図5**のごとく全体を m 個のブロックに分けて道しるべ点 g_i を置き，点 a_0 の座標値が与えられたらまず最寄りの道しるべに飛ぶことで SPP_{a_0} の探索距離を減らすことができる[1]．この場合，各ブロック内の頂点＋交点数を一定値以下にすれば探索時間を $O(1)$ にできる．なお今回は専用の道しるべ点は使わず，ブロック内に既設頂点が在ればそれを道しるべとして使うことにした．ブロック内に図形がない場合には，道しるべは大体目的とする点の近辺に在ればよいから，隣接ブロックの道しるべで代用できる．その場合には入力した図形の頂点をそのブロックの新しい道しるべとする．また N をあらかじめ知ることができない逐次追加の場合に備えて，どれか一つのブロックの頂点数がある値を超えたら全体のブロック分割数を4倍（X, Y それぞれ2分割）に再分割する自動分割方式を採用した．再分割と言っても道しるべ点をブロックごとに設定するだけである．再分割の手数は N に比例するので，図形が極端に偏在する場合を除き1図形あたり $O(1)$ に変わりはない．

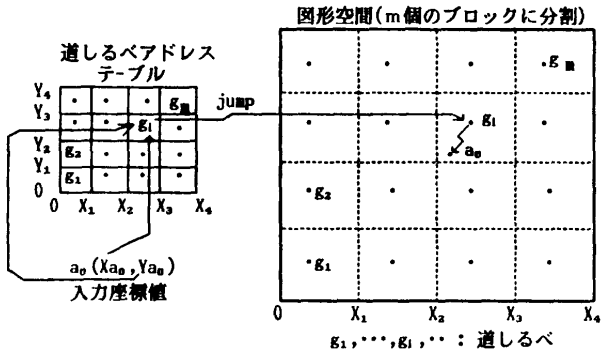


図5 道しるべによる SPP_{a_0} 探索の高速化
Fig. 5 High speed search of SPP_{a_0} using guide post.

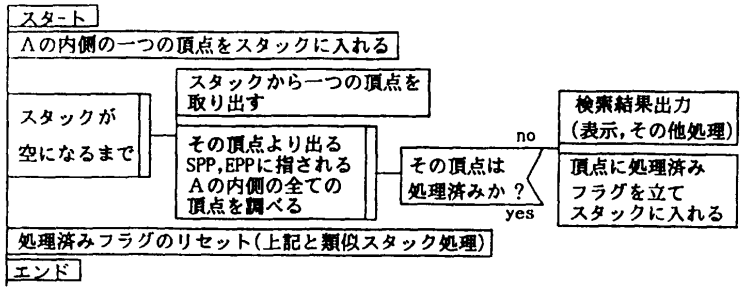


図6 図形 A の内側全頂点データ検索プログラム (PAD[1])
Fig. 6 Program to retrieve all vertices within figure A.

2.3 局所的図形集合演算

対話システムでは処理対象を指定した図形 A の内部に限定した局所的処理にすることが望ましい．そのためには A の外形に包含される図形のみを効率良く検索する必要がある．スペースモデルでは図形 A を指定したときに，A の辺を表現するポインタ EPP にそれが局所処理範囲の外形であることを示すフラグを付加し，その内側の図形のみをすべて検索処理して最後にフラグを取り外す方法を取る．スタックを利用した内部図形の検索手順を**図6**に示す．局所集合演算は，このようにして検索した SPP の空間属性を調べて論理判断すればよい．なお OR, NOT-AND については局所的処理が可能であるが，AND の場合に入力図形の外部をすべて消去することは局所処理ではできない．したがって AND の結果のみを別ファイルに取りだす等の処置をする．また表示用ウインドウについて AND を取ればクリッピング処理ができる．従来のクリッピング手法はウインドウ外部の図形についてもすべてチェックするため処理時間が $O(N)$ のオーダになるが，本方式では $O(1)$ となる．

2.4 重なり図形の表示

標準的ハードウェアで多角形の色塗をすると，内部に包含された位置に在る既表示図形が塗消されてしまうので注意を要する．試作では塗消し防止のため，包含図形は常に最外側から内側へ順に重ね塗するよう順序づけを行った．任意形状に対するこの種の順序づけは普通は大変であるが，SPF では外側図形から SPP を内向きにたどる包含図形検索手法により容易に実現できる[1]．図形 A の内部のみについて塗消し防止の局所的処理をするときには，A の外形から始めて内側へ順に検索処理すれば

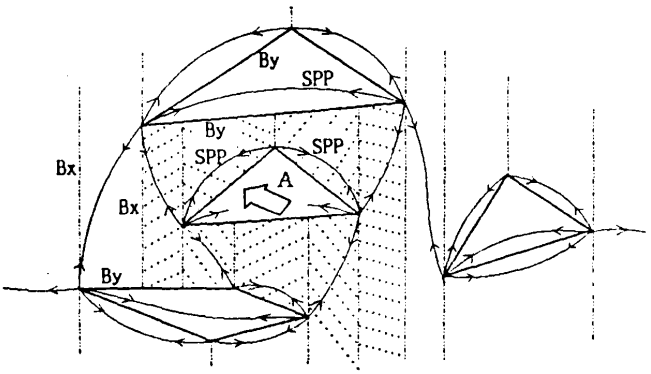


図 7 運動図形 A の衝突検出
Fig. 7 Collision detection of moving figure A.

よい．また A の代わりに表示用ウインドウの外形を使えば，クリッピング付きの処理ができる．

2.5 運動図形の衝突検出

衝突の直前には衝突する二つの図形は必ず一つの隣接空間を挟んで対峙した位置にある．このため図 7 に示す運動図形 A の周りのハッチングした空間の SPP のみを調べれば衝突の相手を局所処理で高速に検索できる．図形 A が運動すると最初は周囲空間の形状がアナログ的に変形して行くが，A の頂点が周囲空間の境界にさしかかると空間の配置にトポロジカルな変化が起こる．もしその境界が衝突相手の図形の外形ならばそこで衝突処理をして終了することになるが，それが頂点を通る仮想線 B_x ならば A はそれを突き切って進むから SPP の更新（付替え）が必要になる．更新を続けながら A を更に進めて行くと，そのうち A の周囲のどこかで衝突が起こる．SPP の更新は A の頂点が B_x を通過するときのみ間歇的に行えばよいから処理は高速である．更新の回数は通過する B_x の数に等しく，したがって処理時間は運動距離にほぼ比例するが，全体の図形数には直接の関係がないから $O(1)$ の処理と言える．

3. 試作プログラムの概要

図 8 に試作したプログラムの構成を示す．試作の目的を基本原理・基本性能の確認・評価のみとし，試作範囲を，二次元 SPF データ構造生成，図形集合演算 (AND・OR)，図形運動と衝突の検出，結果の表示，に絞った．プログラムの規模は C 言語で約 6 k ステップ，使用したハードウェアは日立製作所のエンジニアリング・ワークステーション H-7300 (CPU 68020, コプロセッサ 68881 付，メモリ 4 MB, 1280×1024 高精細度カラーディスプレイ，マウス，マルチウインドウ，GKS 図形インタフェース，UNIX) である．

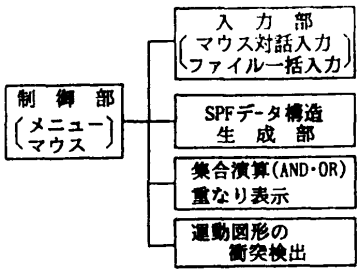


図 8 試作プログラムの構成
Fig. 8 Configuration of the prototype.

システムはマウスによる多角形 1 個ずつの対話入力のほかに，別途作成した多角形群一括入力ファイルを受け付けることができる．入力データは多角形頂点座標列である．システムはこれを受け取ってスペースモデルの SPF データ構造を生成し，マウスによる指示で，注目，集合演算，図形運動，表示を行う．

入力データから生成すべき SPF は，辺を示す EPP，空間を示す SPP の各片側のポインタ EP，SP を並べ，それに頂点座標値を加えた頂点テーブル（図 9）の集合体である．ポインタ SP および EP は，それぞれ相手頂点テーブルの先頭アドレスを示す 4 バイトの整数 p と，相手頂点テーブル内での対応する EP または SP の位置を示す項番（patr の中の 3 ビット）の組合せで表現している．patr の中にはこのほか SP・EP の区別，それらの X 方向を示すフラグ，図形の重なり数を示す SPP の空間属性が設置されている．また vatr は頂点属性で，図形演算処理済みを示すフラグが設置してある．試作システムの許容頂点数（交点含み）は最大 32,000 点．メモリ使用量は空間属性等を含め 1 頂点（交点）あたり 42 バイトであるから，補助テーブル等を含めて全体で約 1.5 メガバイトとなった．ポインタの分だけメモリが多いが，これは機能性能と引替である．なおメモリ使用量は頂点数＋

2バイト	2バイト	2バイト
X	Y	vatr
p[0]		patr[0]
p[1]		patr[1]
p[2]		patr[2]
p[3]		patr[3]
p[4]		patr[4]
p[5]		patr[5]

図 9 頂点テーブルの構成
Fig. 9 Contents of vertex table.

交点数を N として $O(N)$ となる．

4. 性能評価

4.1 多角形入力，重なり検出，表示テスト

写真 1 は 100 個の四角形と 2 個の三角形について SPF を作成し，重なり部を検出して色分け表示した例である．表示色は**表 1** のごとく，赤色が一重かさなり，緑色が二重，青色が三重かさなりに対応している．SPF では重なりの多重度は SPP に空間属性として付加する．試作では SPF 生成プログラムで図形入力ごとにその内部の空間属性に 1 を加え，表示の時に表 1 によって表示するようにした．100 個の四角形の SPF 一括生成時間は CPU 時間で 2.2 秒．マウスによる三角形入力の処理時間は SPF 更新に塗消し防止順序付けと表示を含めて 1.3 秒である．(UNIX の times ( ) 関数で測定)

4.2 図形集合演算

写真 2 は同じ SPF で二重かさなり部のみを表示したもので，図形の AND に対応する．また**写真 3** は OR である．三重 AND 等も同様に可能である．集合演算は SPP の属性を読んで判断するのみであるから高速である．これらの例では処理時間は塗消し防止処理と表示時間を合わせて 2.4 秒である．

4.3 大規模図形の一部に注目，局所処理

写真 4 は LSI のレイアウトパターンを模擬したもので，多角形 1,000 個が重なりあい，頂点と交点の合計数 10,884 個で構成されている．この図形では SPF 生成に CPU 時間で 27 秒，塗消し防止のための順序付け処理と表示に 69 秒を要している．スペースモデルの特徴は各種の処理が局所的に実行できるため，対話処理に適していることである．同じ写真の画面中央上部に見える三角形は，局所処理による対話応答の性能を調べるためのものである．この三角形をマウスにより追加入力すると，システムはマウス位置へ注目点を移動し，SPF へ三角形を追加，その内部のみについて重なりを判定し，塗消し防止をしながら表示を行う．三角形の内部は重なりが増加しているので表示色が変わる．この場合，注目点の移動以外は三角形の内部に限定された局所的処理なので，処理時間は全体図形数によらず高速である．この写真 4 の場合には入力三角形が大きく，これの内部にあって処理すべき既設図形の数が 210 個と多いため CPU 時間は 2.5 秒であった．しかし入力図形が小さい場合には，多角形 1 個をマウスで入力し終わった瞬間に (0.1 sec 程度) 既設図形との重なりを調べた結果を表示してくる．これを上記の全体処理時間 69 秒と対比すれば，対話形における局所的処理の有効性が分かる．

表 1 重なり数と表示色
Table 1 Number of overlaps and color.

重なり数	表示色
0	黒
1	赤
2	緑
3	青
4	シアン
5	黄
6	マゼンタ
7	白

表 2 SPF 上での基本処理時間
Table 2 Processing time on SPF.

SPF 種別 →	(1)写真 4	(2)写真 5
項　目	平均処理時間	
注目 (道しるべ使用)	4.1 ms	4.6 ms
X 側隣接空間検索	200 μs	220 μs
Y 側隣接空間検索	200 μs	160 μs
集合演算	100 μs	120 μs

4.4 注目時間の測定

前記写真 4 と，位置と回転角度を乱数で変化させた 2,000 個の三角形よりなる**写真 5** の SPF の上で，更に別の乱数で与えた目標点に注目するための所要時間を測定した．**表 2** に各 1,000 回測定した CPU 時間の平均値を示す．図形の形状の影響で二つの測定値に多少の差が表れているが，2.2 節で述べた道しるべを使っているので注目時間は原理的に全体の図形数によらず 5 ms 以下と高速である．今回の試作ではこの注目処理をマウスによる図形ピックにも使っているが，全ソフトウェア処理にもかかわらず応答が早く快適である．

4.5 隣接空間の検索時間

前項と同じ SPF 上で同じく乱数で与えた空間に注目してから後，隣接空間を検索するために要する時間を 1,000 回測定し，平均 CPU 時間を表 2 に示した．隣接空間の検索が X, Y 側共に約 200 μs と高速なことがこの方式の大きな特徴である．Y 方向は回転サーチを使って検索している．

4.6 運動図形の衝突検出テスト

写真 6 の水色の人形は運動前の位置である．マウスで人形をピックして運動方向を指示すると，システムは衝突相手の図形を探して衝突地点を計算し，衝突後

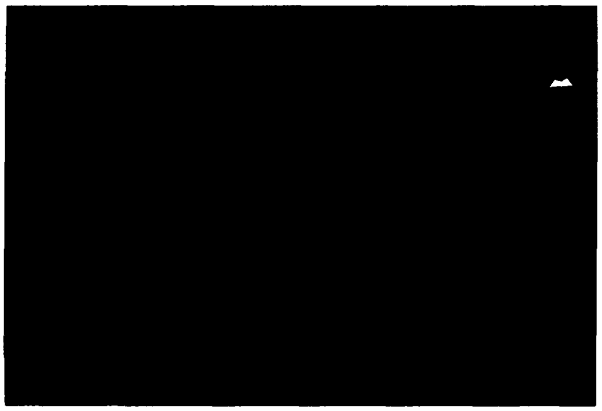


写真 1 重なり図形
赤：一重，緑：二重，青：三重
Photo 1 Overlapped figures.

写真 5 乱数配置三角形 2,000 個
Photo 5 Randomly placed 2,000 triangles.

⬇ 写真 2 図形の AND
Photo 2 Geometrical AND.

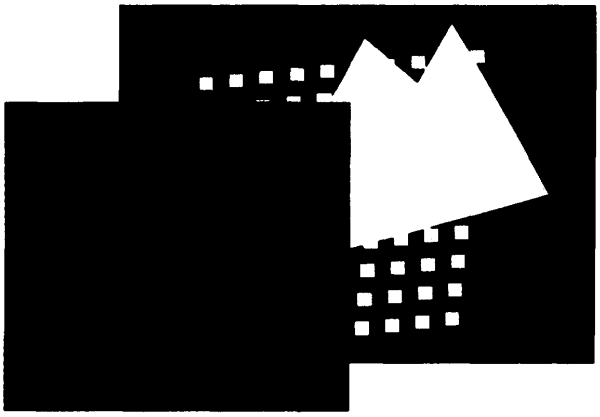


写真 3 図形の OR ⬆
Photo 3 Geometrical OR.

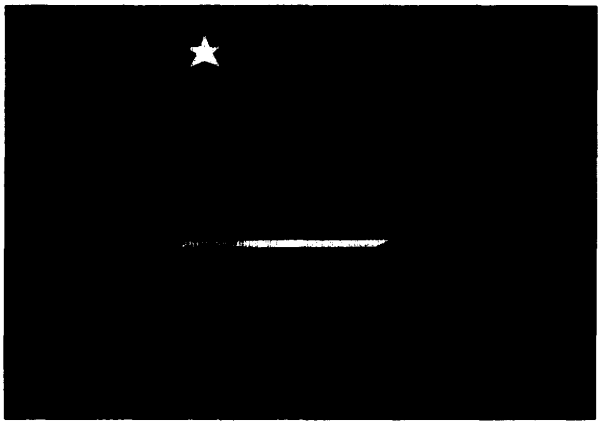


写真 6 運動図形の衝突検出
Photo 6 Collision detection of moving figures.

写真 4 LSI パターンの模擬
Photo 4 LSI like pattern.

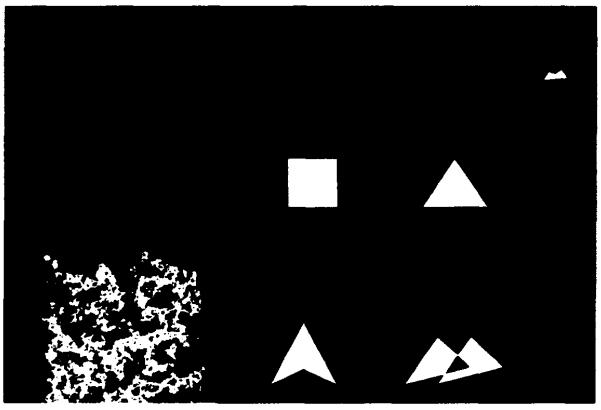


写真 7 特殊形状のテスト
Photo 7 Singular pattern test.

の位置に黄色で人形を表示して停止する．この例では運動させた人形はマウスで入力した 52 頂点の多角形である．また画面の下部には細かい四角形が乱数によってばら撒かれており，画面全体は多角形 2,103 個，頂点と交点の合計数 11,585 個で構成されている．

2.5 節で述べたように処理時間の主要部は運動に伴う SPP の更新時間である．この例では運動と衝突処理の CPU 時間は 220 ms，衝突発生までの SPP 更新回数は 41 回，更新 1 回あたりでは 5.3 ms であった．別の例として，同じ写真上部の白色の星形を画面の最左端から最右端まで動かすテストでは SPP の更新回数 156 回で CPU 時間は 680 ms であった．なお use time は CPU 時間の約 1.2 倍である．

次に人形の下に横たわっている白く細長い四角形を取り去って上記と同じ人形の運動をさせると処理時間が増加して CPU 時間で 5.1 秒になった．これは白い四角形がない場合，その下側にある細かい四角形の頂点を通る Y 軸に平行な仮想的空間分割線 B_x により人形の運動通路が多数の空間に分割され，SPP の更新回数が 980 回（運動経路の B_x の数と運動図形の関連頂点数の積）に増大したからである．これに対し同じ人形を Y 軸に平行に動かした場合は SPP の更新がないので運動距離によらず CPU で約 100 ms と高速であった．CPU 時間 5.1 秒は対話処理としては少々遅いが，非常に細かく空間分割された場所を特に複雑な運動体が通過する用途を除けば実用可能性があると思われる．また SPP を使って運動体（人形）の周辺だけを調べる局所処理を行っているため，全体の図形数が多くなっても処理時間は増加せず他の方式に比べて有利になると言える．

4.7 SPF データ構造一括生成時間

これまで述べたように，いったん SPF のデータ構造ができてしまえば，その上で行う図形処理は高速である．したがって SPF データ構造の初期生成時間が問題になる．**表 3** および**図 10** は各種入力データについて，頂点＋交点の数と SPF 生成時間の関係を実測したものである．なお，NO. 1～NO. 4 の測定では条件を揃えるため，図形数が少ないときには乱数の変動範囲を狭くして，図形重なりの程度を一定にするよう配慮した．また NO. 5 のデータは写真 5 に，NO. 6 は写真 4 に対応したものである．測定結果から，頂点（交点含み）数 30,000 以下の LSI セル程度の図形なら，今回使用したワークステーションでも実用的な性能が得られることが分かった．また第 2.2 節で述べたが，データ構造生成に要する時間は道しるべ方式により原理的に $O(N)$ のオーダになる．これは表 3 の実測値で図形一頂点あたりのデータ構造生成時間 T_s/N が全体図形数によらずほぼ一定であることで裏づけられた．このことから本方式はさらに大規模なデータの処理にも適していると推定できる．

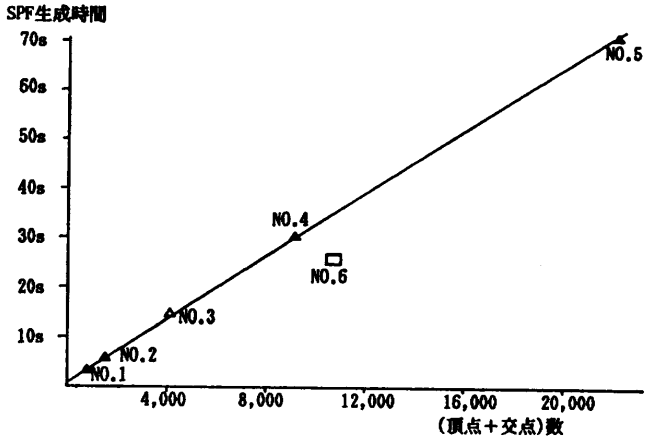


図 10 SPF 一括生成時間
Fig. 10 SPF batch creation time.

表 3 SPF 一括生成時間
Table 3 SPF batch creation time.

入力図形	三角形乱数配置				LSI	
					写真5	写真4
	NO. 1	NO. 2	NO. 3	NO. 4	NO. 5	NO. 6
図形数	100	200	500	1,000	2,000	1,000
N＝頂点＋交点	800	1,650	4,040	9,220	22,597	10,884
T_s＝SPF 生成時間	2.6 s	5.3 s	14.0 s	30.5 s	72.8 s	27.4 s
T_s/N	3.2 ms	3.2 ms	3.4 ms	3.3 ms	3.2 ms	2.5 ms

5. 特殊ケースの検討と対策

5.1 図面最外側空間の処理

図 11 のごとく図面外枠線を考えた場合，SPP_1 の空間の Y^+ 側外形を求める処理では，白矢印 ↶ で示す多数の回転サーチが発生する．このため今回の試作では外枠線を設置せず，最外側空間の SPP に最外側を示すフラグを設置し，このフラグが在るときは外形の検索を行わぬ方式を採用した．ただし，フラグ方式では SPF データ構造生成時間は約 30% 高速化するが，

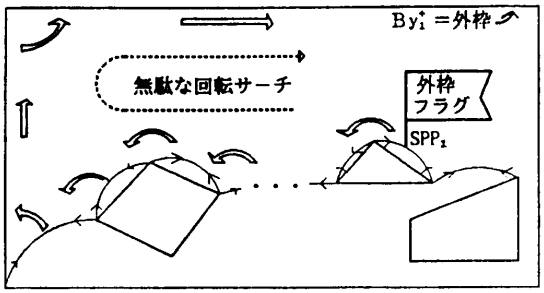


図 11 外枠フラグによる無駄な回転サーチの防止
Fig. 11 Avoiding useless rotary search by SPP flags.

プログラムが複雑化するので再度検討を要する.

5.2 垂直図形と重なり図形の取り扱い

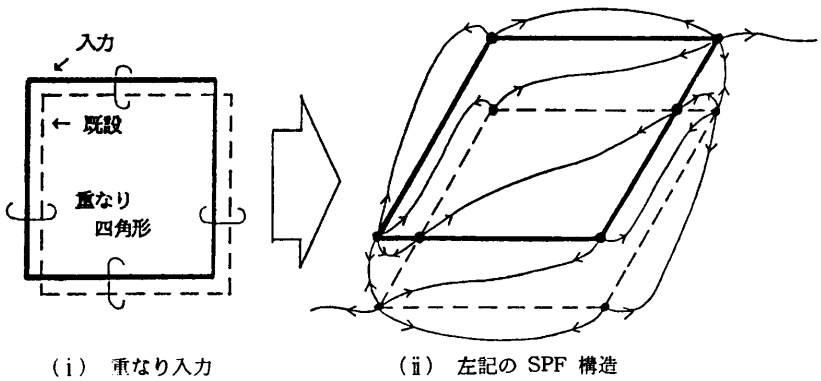
垂直図形や重なり図形については次の方法で特殊扱いの必要をなくしてプログラムを簡単化した.

(1) X 値が等しい二つの点は Y の大きい方が X^+ 側にあると見なす(垂直図形は X^+ 方向にわずかに傾いたものと考える).

(2) 図形が重なる場合には後から入力した図形が Y^+ 側へずれていると考える.

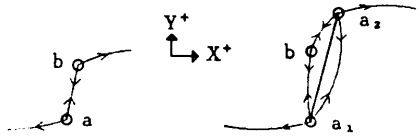
例えば, 垂直・水平辺よりなる完全に重なった二つの長方形は, **図 12** のようにわずかに傾いて Y^+ 側へずれたものとして SPF を作成する. また順次入力された重なった a, b 二点は上記(2)により b 点が Y^+ 側と見なされ, さらに(1)が適用されて結局**図 13** (i) のごとく b 点が X^+ 側にあるとして SPP により結合される.

同じ考え方で既設垂直辺 a_1a_2 に重なる新たな入力点 b 点は辺に対して Y^+ 側にあると考えて図 13 (ii) のごとく設置される. これに上記(1)のルールを適用すると, X 値の等しい3点は Y の値によって a_1, b, a_2 の順に X 方向へ並んでいるとみなされる.



(i) 重なり入力　　(ii) 左記の SPF 構造

図 12 重なり四角形の SPF
Fig. 12 SPF structure of overlapped squares.



(i) 重なった2点　　(ii) 垂直辺と点の重なり

図 13 重なり入力点 b は Y^+ 側へずれていると見なす
Fig. 13 Overlapped point b are considered to be located at Y^+ side.

5.3 重なり点を通る注目点の移動

SPP, EPP をたどって X 方向へ注目点を移動する場合, 頂点座標値の比較によって進行方向を判定する方法では, 頂点の重なり点で座標値が等しくなるため進行方向の判定ができなくなる. しかし重なり点については前項(1), (2)によって SPF 生成時に X 方向の並び順が決められているので, それらを結合する SPP, EPP 自体に属性として X 方向の情報を持たせ, これにより進行方向を決めればこの問題は解決する.

5.4 垂直辺, 重なり図形処理の例

写真 7 は垂直または重なりのある頂点や辺に関する動作安定度テストの例である. 図形の色は第4章表1により図形の重なりを示す. したがって例えば画面右側の白色三角形では, 図形が7重に重なっているがシステムが正常動作していることを示す. 画面右下の2個ずつ交叉した三角形は, 交叉点が多重に重なる場合のテストである. 中央下は, 多重重なり図形の頂点がさらに同一座標位置にある場合のテスト. 左下の大きな四角形は一辺の長さがこの二分の一の四角形を, そのまた二分の一ずつずらして X, Y 方向に各3個ずつ合計9個並べたもので, これも辺と頂点の重なりテストである. 最後に左下は, 900 個の三角形の位置と回転を乱数で変化させて重ね, 予期しない計算誤差や特殊ケースに対する強さを調べたものである. ここでは約 25,000 個の交点が発生している.

以上各種の対策を行った結果, 普通に少しテストした程度では問題を起こさないプログラムになった. しかし, 辺と頂点と計算誤差を持つ交点が複数個同一座標点で重なる特殊ケース等では, まだまれに誤動作を起こす場合がある. これについて

は今後さらに検討を続ける予定である.

6. まとめ

二次元スペースモデルについて原理試作を行った. 試作範囲はデータ構造生成, 図形集合演算, 運動図形の衝突検出, および表示に絞ったが, 基本原理・基本性能の確認はできたと考えられる. 以下, 確認した事項を列挙する.

(1) スペースモデルのデータ構造の生成が原理どおり可能である. 生成時間は1項点 (または交点) あたり 4 ms 以下, 全体 N 個の頂点 (交点) に対し $O(N)$ であり, 実用的にも高速であることがデータで裏づけられた.

(2) 上記データ構造の上で, 一部に注目する局所的処理が可能である. 対話形図形集合演算は表示図形が小さければ応答は 0.1 秒程度と瞬間的で, 高性能と言える.

(3) 同じデータ構造の上で隣接空間の検索は約 200 μs, 重なり空間の検索は約 100 μs で実行できる.

(4) 運動図形の衝突検出が対話速度で可能である. 応答時間は衝突までの移動距離 (ポインタ更新回数) にほぼ比例するが, 短距離なら 0.1 秒, 複雑長距離でも5秒程度と実用可能範囲である.

(5) 演算結果の表示においては, 包含図形の塗消しを防止する効率の良い順序づけが可能である.

(6) 辺や頂点の単純な重なりについては, それらが相互にわずかにずれていると考える取扱いが可能である.

(7) 頂点と辺と交点が同一座標で複数個重なるような極端な形状ではまれに計算誤差による問題を起こす場合があり, さらに検討を要する.

今後の課題としては次のものが残されている.

(i) 計算誤差問題の解決

(ii) 隠面/隠線消去, LSI CAD 等への応用検討

(iii) 三次元への展開, 等

謝辞 終わりに今回の試作にあたりご支援くださった日立製作所マイクロエレクトロニクス研究所猪瀬文之所長, 遠藤裕英部長, 松田泰昌主任研究員ほか関係各位に感謝の意を表します.

参考文献

1) 大沢 晃: 二次元スペース・モデリングの考察, 情報処理学会論文誌, Vol. 27, No. 12, pp. 1174-1185 (1986).
2) 大沢 晃: 三次元スペース・モデリングの考察, 情報処理学会論文誌, Vol. 27, No. 12, pp. 1186-1196 (1986).
3) Ousterhout, J. K.: Corner Stitching: A Data Structuring Technique for VLSI Layout Tools, *IEEE Trans. on CAD*, Vol. CAD-3, No. 1, pp. 87-100 (1984).
4) Ousterhout, J. K. et al.: MAGIC; A VLSI Layout System, *21st Design Automation Conf. Proc., '84, IEEE ACM*, pp. 152-159 (1984).
5) Tukizoe, A., Sakemi, J., Kozawa, T. and Fukuda, H.: MACH; A High-Hitting Pattern Checker for VLSI Mask Data, *ACM IEEE 20th Design Automation Conf. Proc.*, pp. 726-731 (June 1983).
6) Carlson, E. C. and Rutenbar, R. A.: A Scanline Data Structure Processor for VLSI Geometry Checking, 情報処理学会 CAD エンジン講習会資料 (June 1987).
7) Lee, D. T.: Computational Geometry—A Survey, *IEEE Trans. on Comput.*, Vol. C-33, No. 12, pp. 1072-1101 (1984).
8) 鈴木則久: VLSI CAD への計算幾何学の応用, 情報処理, Vol. 24, No. 4, pp. 563-566 (1983).
9) 二村良彦ほか: PAD (Problem Analysis Diagram) によるプログラムの設計及び作成, 情報処理学会論文誌, Vol. 21, No. 4, pp. 259-267 (1980).

(昭和62年8月3日受付)
(昭和63年9月5日採録)

大沢 晃 (正会員)

昭和11年生. 昭和34年名古屋大学工学部電気工学科卒業. 昭和36年同大学大学院応用物理修士課程修了. 同年(株)日立製作所入社. 日立大みか工場で制御用計算機ハードウェアを設計. 主としてプロセス入出力装置, A-D コンバータ, 磁気ディスク, CRT 等, アナログ関係を担当. 昭和50年より日立武蔵工場で半導体用 CAD を開発. 昭和63年日立京浜工業専門学院主任教授, 現在に至る. 電子情報通信学会会員.

川崎 敏治

昭和36年生．昭和59年，東京理科大学理工学部数学科卒業．昭和61年，同大学院修士課程修了．同年，(株)日立製作所入社．現在までに，同社マイクロエレクトロニクス機器開発研究所において，スペースモデル，文書処理システムなどの研究開発に従事．現在，同研究所勤務．

岩本 哲夫（正会員）

昭和21年生．昭和44年大阪大学基礎工学部制御工学科卒業．昭和46年同大学院修士課程修了．同年(株)日立製作所入社．同社研究所にて，ダイヤ作成システム，物流システム，CAD システム，文書処理システムなどの研究開発に従事．現在，同社マイクロエレクトロニクス機器開発研究所に勤務．電子情報通信学会会員．