二次元スペース・モデリングの考察†

——ポインタ方式空間表現による局所的図形処理——

大沢 晃††

LSI，地図，等大規模図形処理を効率化するためには，図形の必要部分に注目した局所的処理を行うことが望ましい．本論文では，これを実現するため図形空間をポインタで表現する新しい図形データ構造を提案し，その原理・応用・生成手法等につき考察する．本方式によれば図形処理の基本となる隣接空間や重なり図形の検索が，例外的な場合を除き局所的に可能になり，処理時間は総図形頂点数 N^* と無関係で $O(N^0)$ のオーダとなる．その結果，局所的図形論理操作 (AND, OR, 等)，運動図形の衝突検出，注目点近傍の探索，さらに図形データベースのメンテナンス，等が例外的な場合を除き $O(N^0)$ の時間で実現できる．また本方式は，点，折れ線から，凹凸，斜辺，穴を含むステンドグラス風多色図形までを取り扱うことができる．これらの機能は，LSI レイアウト図の対話形デザインルール・チェック，同コンパクション，地図における道順探索，等従来困難であったり，長時間を要するとされていた処理のいくつかを実用的な速度で実現可能にするために役立つと考えられる．

1. まえがき

VLSI 用 CAD，地図処理システム等大規模図形を取り扱うシステムでは，従来手法では計算時間が図形規模に依存して増加するため原理的には可能でも実用上は不可能に近い大きな問題がたくさんあった[1),2),4)]．その一つの原因は本来この種図形処理が図形相互間の重なりや隣接図形間の関係を処理するものであるにもかかわらず，図形データベースにこれらの情報が記述されていないため，必要図形のみの効率的検索ができず，関係のない遠方の図形データまで読み出してきて無駄な処理をしていたところにあると考えられる．

この問題の対策として Ousterhout ら[3)]は図形空間を「Tile」と名付けた部分空間に分割してポインタで接続する「corner stitching」と呼ぶ図形データ構造を提案している．しかしこの方法は図形形状が垂直・水平方向の辺よりなる矩形**の組合せに限定されるため，汎用性の面で大きな問題を残していた．

今回提案する方法は部分空間をポインタのペアで表現する新概念（ポインタ方式スペース・モデリング）に基づき，点，折れ線から凹凸，斜辺，穴明き，重なりを含むステンドグラス風多色図形までを取り扱え，かつ局所的図形処理を可能とする図形データ構造とその応用に関するものである．本方式によれば各種形状を含む大規模図形において，隣接空間や重なり図形の検索が，付録2に示す例外的な場合を除いて局所的に実行できる（図1）．この応用により，必要部分のみに注目した局所的図形論理操作 (AND, OR 等)，運動図形の衝突検出，近傍の探索等の処理が $O(N^0)$ (総図形頂点数 N^* に依存しない) の時間で実現する．

最終的な用途としては，斜図形を含む LSI レイアウト図の対話形デザインルール・チェック，同コンパクション，地図における道順探索等が考えられる．また本方式では，大規模図形データの中から一部分を切り出すウインドウイングや，マウスなどで指示された図形のピックアップが全体図形を調べることなく可能であり，データベースへの図形の追加，削除も $O(N^0)$ の

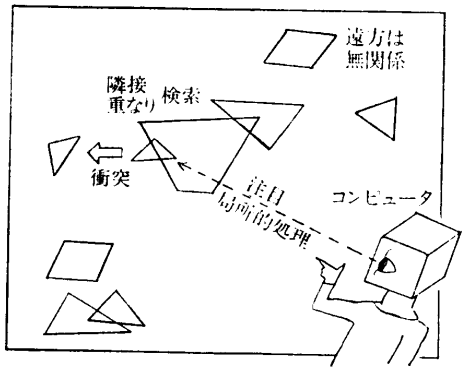


図 1 隣接，重なりの局所的検索による高速図形処理
Fig. 1 High-speed processing by local retrieval of adjacent/overlapped figures.

† Pointer-based 2D-Space Modeling Method Allowing Local Geometric Processing by AKIRA OHSAWA (Musashi Works, Hitachi Ltd.).

†† (株)日立製作所武蔵工場

* 図形全体の規模の指標として，全図形の頂点数の総和 N を用いる．

** 台形への拡張は可能である．しかしこの手法は空間は表現するが原理的に点や線の表現を持たぬので，それらを含む図形への拡張はそのままでは困難と思われる．

時間で実行できるので，オンライン対話処理に適していると言えよう．

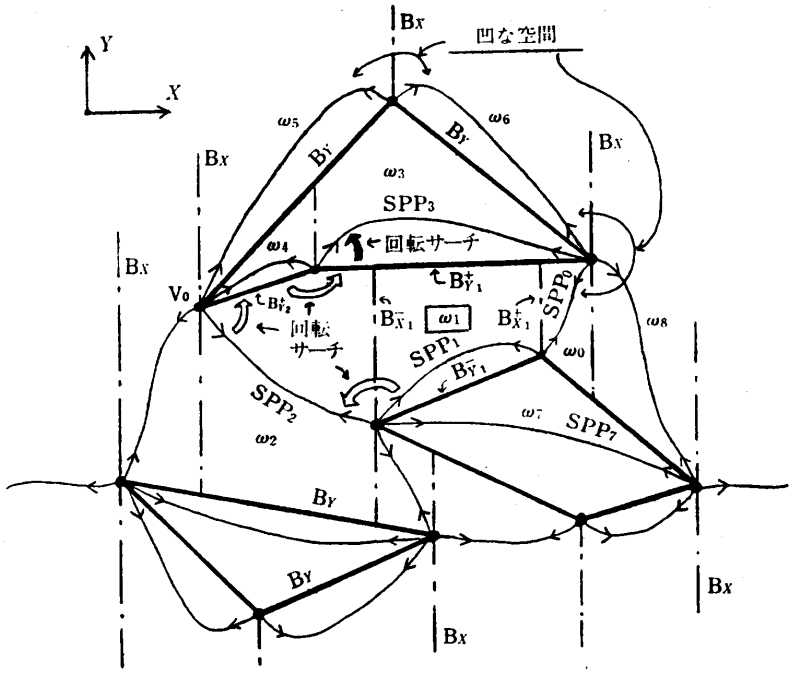
以下第2章では基本原理，第3章では応用範囲と性能，第4章ではデータベース生成手法について述べる．

2. ポインタによる図形空間モデル化の概念

2.1 部分空間のポインタによる表現

ここでは隣接図形検索のための準備を行う．**図2**に示すように，頂点の周りに凹な（辺に挟まれた角度が180°より大な）空間があれば，それを頂点を通り Y 軸に平行な仮想的境界線 Bx（同図1点鎖線）で区切り，これと各図形の辺 By によって図形空間を部分空間 $\omega_0, \cdots, \omega_i, \cdots$ に分割する．この場合 Bx が Y 側の隣接 By を突き抜けることはないとする．また Bx は仮想線であるから，隣接 By との交点には交点情報は設置しない（データ量とそのメンテナンス処理の増大防止）．空間を頂点を通る境界線で区切ったのであるから，各 ω_i はその X^+, X^- 端に必ず頂点を持つ（辺と辺の交点も頂点とする）．その頂点にそれぞれ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）と呼び，SPP を設置した全体の図形データ・ファイルのことを SPF(Space Pointer File) と名付ける．

SPF の空間分割は図形データと座標系が決まれば一義的に定まるので，任意の部分空間 ω_i について隣接部分空間は一義的に定まる．また，X 座標値が等しい頂点や，重なった辺が存在する場合には，**図3**(ⅰ)，(ⅱ)のように一方がわずかにずれていると考えることにすれば，どの空間にも左右端にはそれぞれ必ず1個で，かつ1個に限る頂点が存在する（$X=\pm\infty$にも頂点があるとする）ため，各 ω_i には必ず一つの SPP（これを SPP_i とする）が1対1で対応することになる（付録1参照）．1対1なので，SPP_i が与えられた



● SPP_i が与えられたとき，ω_i の外形，隣接空間を局所的に検索
（1） X 側外形 $B^+_{X_1}$, $B^-_{X_1}$，隣接空間 SPP_0, SPP_2 は，頂点経由ポインタで検索
（2） Y 側外形 $B^+_{Y_1}$，隣接空間 SPP_3 は回転サーチ ↶，↷ で検索

図2 SPF における部分空間 ω_i の $SPP_i(i=0,1,2,\cdots)$ による表現と隣接検索
Fig. 2 Representation of ω_i by SPP_i, and retrieval of neighborhood.

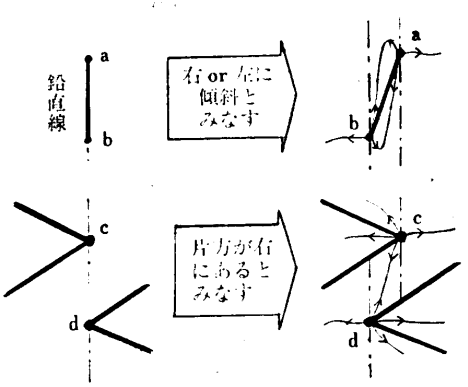


図3(ⅰ)　X 座標値の等しい図形
Fig. 3(ⅰ) Figures have the same X-coordinate.

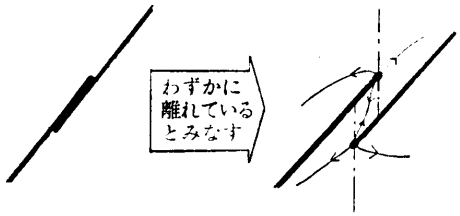


図3(ⅱ)　重なり線
Fig. 3(ⅱ) Overlapped lines.

とき SPP_i から対応する ω_i の情報を求めることが容易に可能なら，SPP_i は部分空間 ω_i を表現していると言ってよい．SPP_i が ω_i を表現しているならば，各種図形処理における図形（空間）検索問題は SPF の中から必要な SPP_i を検索する問題に置き換えることができる．次節ではそのために，SPP_i から ω_i を高速に求める手法等につき説明する．なおここでは，図形外部の空間も内部と対等な空間として陽に表現されることに注意したい．

2.2 空間外形と隣接空間の局所的処理による検索

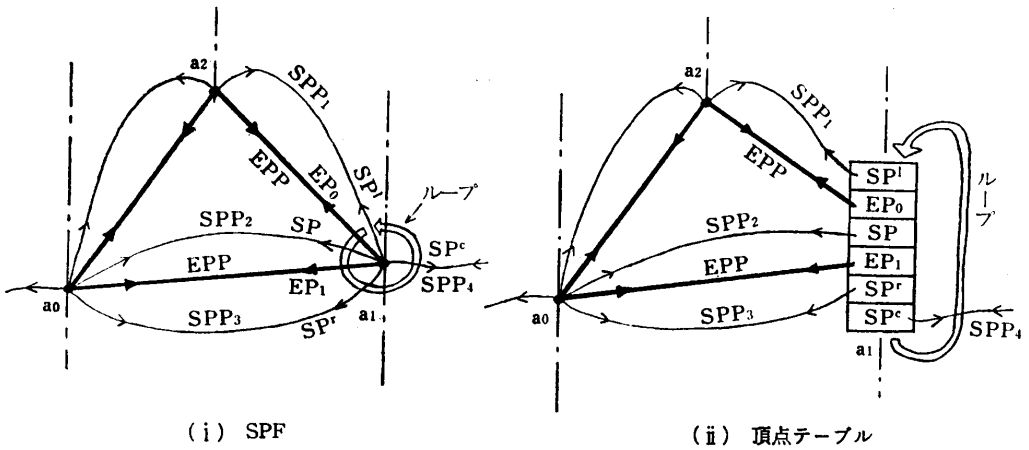
本節では，SPF 構造の図形データベースが作成されているとして（作成法は第4章で述べる），SPP_i が与えられたとき対応する ω_i の外形線およびこれに接する隣接空間を例外的な場合を除き $O(N^0)$ の処理時間で検索する手法につき述べる．

まず，図2で SPP_1 が与えられたとしたとき，ω_1 の X^+ 側外形線 $B^+_{x_1}$ と隣接空間 ω_0 を表現する SPP_0 の検索は，図から明らかなようにポインタを SPP_1 から X^+ 方向へたどることにより直接的に可能である．この処理は遠方の図形とは無関係な局所的処理であるから，総図形頂点数 N とは無関係な $O(N^0)$ の処理である．

次に ω_1 の Y^+ 側外形線 $B^+_{y_1}$ を求めるには，同図 SPP_1 からスタートして白矢印 ↶ で示すように頂点の周りで隣り合って接続している SPP または図形辺を，一方向へ回転するようにたどりながら検索すれば必ず求める辺 $B^+_{y_1}$ に到達できる．この処理を回転サーチと名付ける．回転サーチにおいて必要な頂点の周りで隣り合うポインタの検索は，図4に示すように頂点に設置するポインタを周回順にループ状に並べた頂点テーブル上で，隣り合うアドレスを調べることで容易に実行できる．SPF の図形データは具体的にはすべてこの頂点テーブルで表現する．この方式では辺も SPP と同様に頂点間で指し合うポインタ EP の組 EPP で表現しているから，図2で回転サーチが SPP_2 から $B^+_{y_2}$ に移るような場合にも，ポインタの種類（EP, SP）を意識することなく回転サーチの処理を行うことができる．なお図4では頂点座標値 (x, y) や，ポインタの種類（EP, SP）を示す属性等は図示を略してある．

頂点テーブルの EP, SP の並び順は，その頂点の周りの辺と空間の配置を表現しており，他の図形の影響は受けない．したがってある図形の追加・削除等は周囲空間の分割変化によって他図形頂点テーブルの EP, SP の指し先を変化させるが，EP, SP の並び順やテーブルの長さに影響を与えることはない．これは頻繁なメモリ管理操作によるプログラム効率低下を防ぐ意味で効果的である（頂点テーブルの長さは，辺数等頂点自体の形状が変われば変化する．また図形辺の交叉の発生・消滅に対しては交叉点の頂点テーブルを設定・解放する必要がある．しかし，その頻度は普通には比較的少ないと思われる）．

さて図2に示すように，SPP_1 を与えられて外形辺 $B^+_{y_1}$ を求める回転サーチは，SPP_1 から出発して $B^+_{y_1}$ またはその延長辺に接している部分空間の SPP を，その辺上，少なくとも図形の X^- 側端には必ず存在する頂点 V_0 に達するまでたどり（図2では $SPP_1 \rightarrow SPP_2 \rightarrow V_0$），そこから逆に目的の $B^+_{y_1}$ に達するまで辺に沿って戻る限定された空間内だけの局所的処理である．したがって遠方の図形とは関係がないから，付



（i） SPF　　（ii） 頂点テーブル

図 4 SPF（i）の頂点テーブルによる表現（ii）
Fig. 4 Representation of SPF (i) by vertex table (ii).

録2に示すような例外的な場合でなければ，総図形頂点数 N と無関係な $O(N^0)$ の処理であると言える．なお $B^{\downarrow}_{Y_1}$ への到着は辺の両端の X 座標値から判定できる．また $B^{\downarrow}_{Y_1}$ の Y^+ 側空間 SPP_3 の検索は $B^{\downarrow}_{Y_1}$ からさらに Y^+ 側へ ↰ 印の回転サーチを行えばよい．$B^{\downarrow}_{Y_1}$ 等 ω_1 の Y^- 側の検索も同様である．

以上で与えられた空間の上下，左右の外形と隣接空間の検索は，付録2に示すような例外的な場合を除けば，局所的処理で実行できることが分かった．この検索を繰り返せば，図形空間上で隣接空間伝いに任意の方向へ自由自在に注目点を移動させながら必要な処理ができることになる．これは本方式の大きな特徴である．

2.3 与えられた点 a を含む部分空間の検索

前節では計算機の当初に注目すべき空間 ω_i に対応する SPP_i が最初に与えられていたとして，その外形 (B_X, B_Y) や隣接空間を求める手法を説明した．本節では図5において，例えばマウス，タブレット/ペンなどにより点 a の座標値 (x_a, y_a) が与えられるが，点 a がどの部分空間に含まれるかは与えられていないとき，点 a を含む部分空間 ω_a に対応する SPP_a を SPF の中から探し出す方法，すなわち計算機に SPP_a に注目させる手法について述べる．

そのためには前節で述べた任意方向隣接空間の検索機能を利用する．すなわち，最初に図5の既設 SPF の中から任意の一つの SPP^0 をとり，SPP^0 の両端の頂点の X 座標値を x_a と比べる．もしその範囲が x_a を含んでいなければ，そこから x_a を含む空間に達するまで，かつなるべく y_a にも近い方向へ，x_a 方向の SPP または EPP をたどって接近する（図5 ➞ 印）．接近を進めて X 座標値の範囲が x_a を含む SPP_{x_a}（または EPP_{x_a}）に達したら，今度は回転サーチによって，y_a 方向でかつ x_a を含む隣接空間だけを順にたどってゆけば必ず SPP_a に到達する．この手法は探索ルートが最初は X 方向に，次は Y 方向に一方向的であるため，ループや振動はあり得ず，また全空間調査もしないので効率がよい．

SPP_a 探索時間の平均値は，図形分布が均等で，かつ付録2に示す例外的な場合が無視できれば，任意に

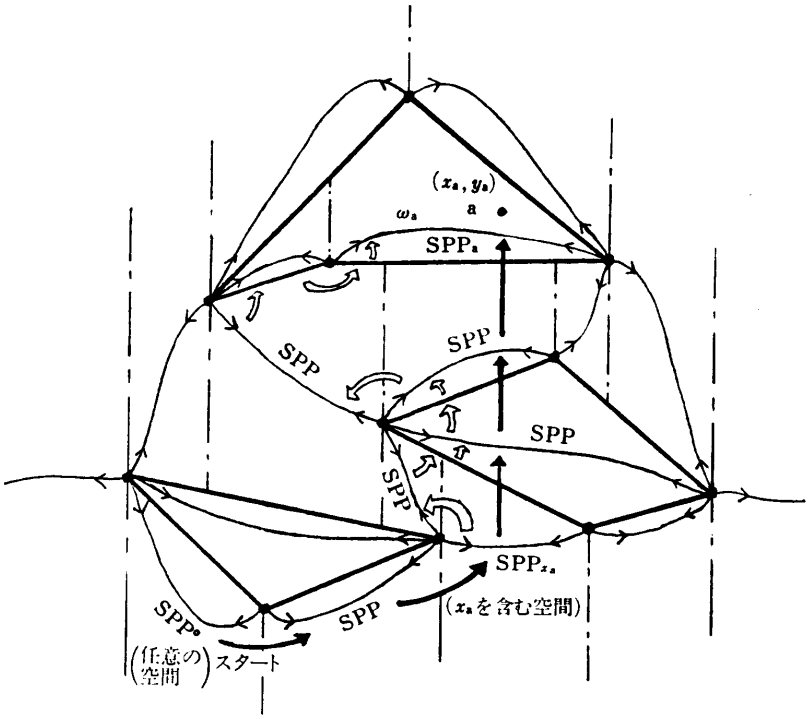


図5　点 $a(x_a, y_a)$ の位置する空間 SPP_a の検索（太矢印 ➞）
任意の SPP^0 からスタート，SPP_{x_a} 以降の Y 方向検索は回転サーチ（↰ 印）による
Fig. 5　Retrieval of subspace SPP_a, including point $a(x_a, y_a)$.

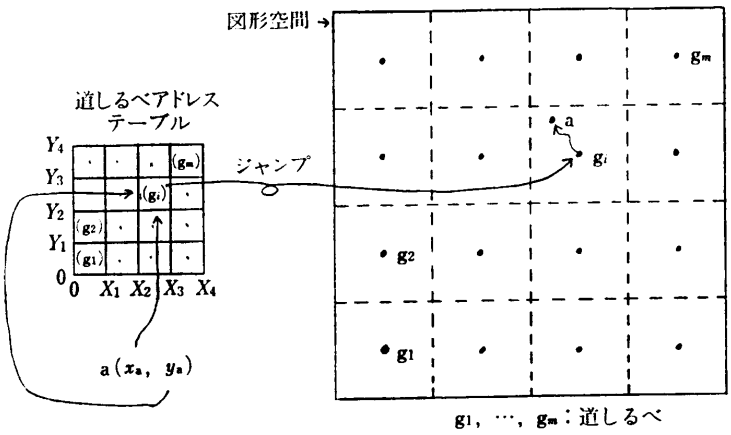


図6　道しるべ g_i 導入による SPP_a 探索の高速化
Fig. 6　Speeding up of SPP_a search by introduction of guide post g_i.

選んだ SPP_0 から SPP_a までの平均距離に比例するから，オーダとしては $O(N^{1/2})$ となる（全頂点数 N の図面の中の直線距離）．

さらに高速化するには，**図6**のごとく図面全体を m 個 ($\propto N$) に区分し，各区間に道しるべ点 g_i ($i=1,2,\cdots,m$) をあらかじめ設置し，a の座標値が与えられたらその区間の g_i へ直接ジャンプし，そこから上記手法で SPP_a を探索する手法をとる．区間数 m を N に比例してとり，一区間内の頂点数がほぼ一定にでき，かつ付録2の特殊ケースが無視できれば，平均探索時間は $O(N^0)$ に抑えられる．一種のバケット・ソートである．例えば，一区間の頂点数が100程度になるよう区分すれば，g_i からの平均探索距離は大体10以下となり，かつ g_i 設置によるメモリ増加量は1%程度（100頂点につき g_i 1頂点追加）で実用範囲と思われるが，図形の偏り等性質によるのでその都度検討を要する．以上の処理をまとめて**図7**に示す．

2.4 空間属性保持による重なり検索の高速化

図8(i)において与えられた図形Aと他の図形との

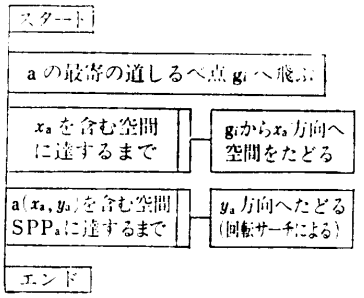
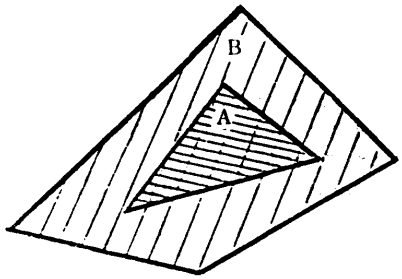
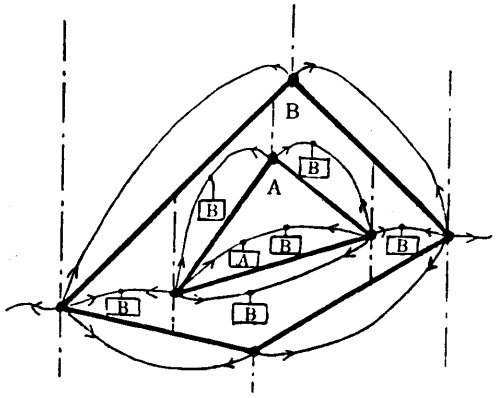


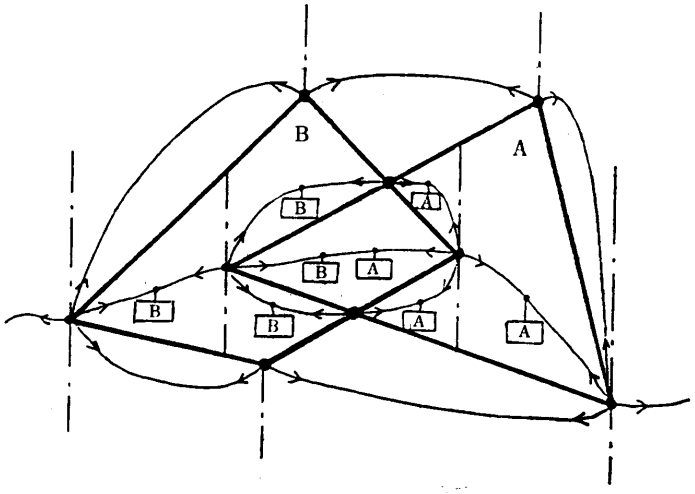
図7 点aを含む空間 SPP_a の検索 (PAD[6])
Fig. 7 Retrieval procedure of SPP_a, subspace including point A.



(i) 色による重なり判定
(i) Cognition of overlapped figures by color.



(ii) 空間属性（凸凹）による重なり判定
(ii) Cognition by space attributes (凸凹).



(iii) 交叉図形の場合
(iii) On the crossed figures.

図8 空間属性による重なりの表現
Fig. 8 Representation of overlapped figures by space attributes.

重なりを調べる問題を考える．すべての図形の内部が各図形ごとに独得の色で塗られており，図形の重なりを色の重なりとして見ることができるとすれば，重なりの判定はAの内部の色の重なりを見るだけで十分であり，図形の形状や数に無関係な高速判定ができるはずである．これを計算機で実現するために SPF を利用する．

SPF ではポインタの組 SPP によって部分空間を表現しているので，図形内の部分空間に色を塗る代りに SPF 作成時に図形内部を表現する SPP に色に相当する空間属性（図形名等）を付加しておく．そうすれば必要なときに図形A内部の SPP を調べるだけでAがどの図形と重なっているかを直ちに知ることができる（図8（ⅱ），（ⅲ））．従来の手法のように辺の交叉を一つずつ調べる必要が全くないので，SPF さえできていれば図形が複雑大規模なときにも重なりが直ちに判明し，処理時間は $O(N^0)$ である．

SPF への EP, SP の設置は多少複雑なので別途第4章で説明するが，空間属性の付加は追加図形内の SPP に，追加図形の空間属性と追加前の下地空間の属性を加えて付加するだけでよく容易である．この場合の処理は追加図形内部に限定されるので，例外的な場合を除き処理時間は1図形当り $O(N^0)$ となる．

2.5 SPF 表現の汎用性について

SPF 構造は図1に示したポリゴンのほかにも各種図形に適用可能である．**図9**は折れ線の端点や孤立点も一種の頂点として空間分割を行うことにより，同様な取扱いが可能なことを示している．特に**図10**のように SPF 中の図形がすべて孤立点ばかりである場合には，各点は X 座標値の順に SPP で接続され，ちょうど X 座標値でソートされた形になる．この意味で SPF は従来のソーティングの二次元有限寸法図形への拡張形であると考えることもできる．

図11はステンド・グラス風の多色図形が SPF で取り扱い得ることを示す．各部の色彩は SPP に付加する属性で表現される．**図12**は SPP の属性を ϕ（空）にすることで，図形の穴を表現する方法を示す．これによれば図形の外形が与えられたとき，穴の有無を調べる処理は SPP を内側へたどることにより高速に実行できる．内側の認識は SPP に付加された空間属性によればよい．

図13に示す曲線についても，図形の X^+ 端，X^- 端に一種の頂点を設置することにより基本的には同様の取扱いをすることができる．なお本件については付録1の一般的な空間分割法を参照されたい．

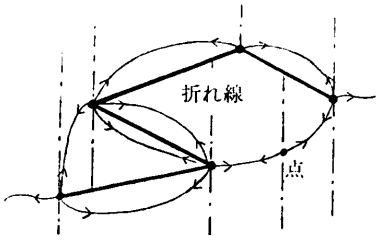


図9 折れ線，点
Fig. 9 Folded line, point.

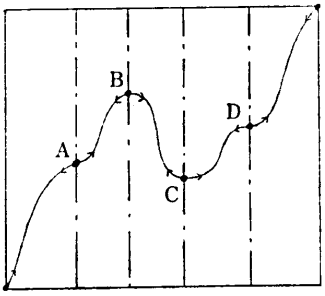


図10 孤立点の SPF は X 値によるソートと同じ
Fig. 10 SPF of isolated points stands for sorting by X-coordinate.

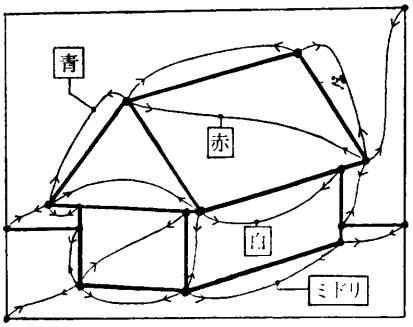


図11 多色図形
Fig. 11 Multi-colored figure.

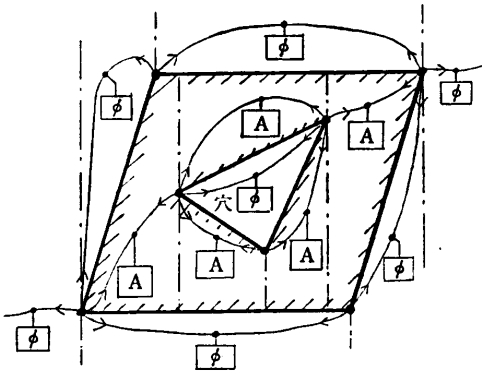


図12 穴のある図形
Fig. 12 Figure with hole.

CPU の主メモリに入り切らない大規模図形データでは，**図 14** のごとく全体を複数のページに分割してディスクに格納し，必要ページのみを主メモリに呼び出して処理する．このときページ間を渡る SPP によってディスク・アクセスが多発するのを避けるためには，同図に示すように各ページの外形線をエンド・ユーザには見えない線図形として設置すればよい．この方法によれば（そこで空間が切断されるので）SPP はすべてページ内で終端してしまうから，ページ内処理のときに SPP を検索しても処理が主メモリ外の他ページへ飛ぶことが無くなり処理効率が向上する．なお，同図 c_1c_1', c_2c_2' のように図形辺がページ間を渡るときには接続用テーブルを設けて相手を指示する等が必要である．また大規模図形については外形線で囲まれた内部の詳細をセルまたはモジュールとして取り扱う階層構造化が重要であるが，本論文の主旨から外れるので省略する．

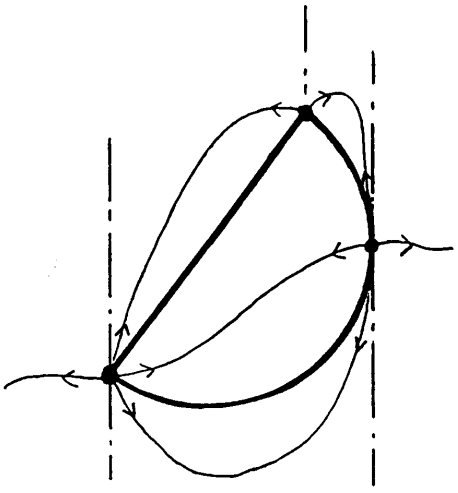


図 13 曲線
Fig. 13 Curved line.

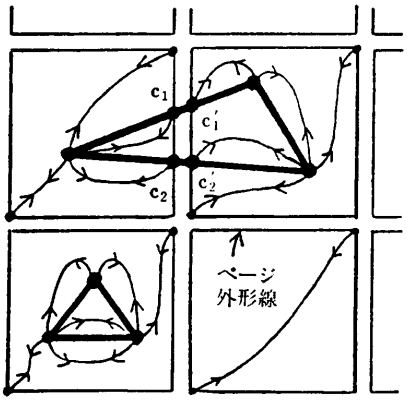


図 14 大規模データのページ分割
Fig. 14 Partition of large scale data into multiple pages.

3. SPF の応用

スペース・モデルは図形空間の持つ基本的な性質（隣接，重なり）を正確に表現しているため，SPF 構造ができていれば，各種の用途に一貫して効率的に対処できる点に特徴がある．本章ではこれらの応用につき概観する．なお，前提となる SPF の生成を高速に行う手法については第 4 章で述べる．

3.1 局所的図形論理演算 (AND, OR, 等)

図形の重なりはデータベース SPF 作成時に SPP の属性として付加してあるから，例えばある図形が与えられ，それと他のすべての図形との間の **AND** 処理が要求されたら，与えられた図形の内部にあるすべての SPP の属性を調べて，重なり部の図形だけを取り出せばよい．与えられた図形 1 個当りの処理時間は，一般には総図形頂点数 N とは無関係で $O(N^0)$ となる（与えられた図形内部の空間数が N と無関係と仮定）．

図形の **OR** 処理は図形データから重なり部分の辺を消去し，SPP の属性を書き換えればよいし，**EOR** は重なり部 SPP の属性を空にすればよい．また図形の切削加工 (**NOT AND**) は SPF から与えられた図形との重なり部のみを削除すればよい．これらはいずれも **AND** と同様一図形当り $O(N^0)$ の部分処理で，オンライン化に適している．

3.2 局所的処理による高速ウインドウイング

大規模図形から表示用ウインドウ内部に当たるデータだけを切り出す処理も，局所的処理で高速に実行できる．最初に切り出すべきウインドウの外形線を SPF に入力し，次にその内部にあるすべての図形を SPP をたどりながら抜き出し，最後に当初入力したウインドウ外形線を削除すればよい．この場合，ウインドウの外側のデータを調べることは原理的に不要なので（ウインドウ外形線に接する空間については調べるが，それは全体に比べ一部分にすぎない），総図形頂点数 N に無関係で，従来のいわゆるクリッピングより高速，かつ切り抜き形状が任意という特徴がある．

3.3 注目，近傍の探索，図形認識への応用

前節までで述べたように，SPF を使えば大規模図形の一部分に注目して，必要な図形処理をその付近だけで局所的に実行できる．例えば LSI のレイアウト等では図形の対話入力，修正に伴って必要図形だけについ

てのデザインルール・チェックがオンラインで実行できるし[3),5)]，地図に適用すれば道順の自動探索等にも応用できる可能性がある．

図形認識への応用については，部分への注目能力のほかに，図形相対位置関係の把握能力，すなわち上下，左右，隣接，重なり，包含，等の認識，近傍図形をグループ視する能力等，人間は簡単にやってのけるが従来計算機では能率が悪すぎると言われていた処理が高速度で実行できるようになる．実際，例えば**図 15** の中から ⧋ と △△ を探し出すような問題は，図形数が多くなると従来手法ではかなり難問であるが，SPF 構造ができているとすれば，□の中に△があるか，△の横に△があるか等を SPF 上で 1 個ずつ調べることができる（ここでは□と△の形状は，辺を一巡して頂点数を数える等の方法で容易に識別できるとする）．目的のパターンがありそうな場所を教えられたら，その付近を優先的に調べるとか，1 個見付けたらそれで調査を中止するような人間と類似の動作も可能である．図形 1 個当りの調査時間は（SPF 生成時間を別にすれば）$O(N^0)$ であり，全体を調べても $O(N)$ 程度である．

3.4 運動図形の衝突検出

静止図形間の干渉チェックは従来から多く行われていたが，運動図形の衝突検出は簡単ではなかった．干渉チェックがスタティックであるのに，衝突検出は時間を含むダイナミックな処理であり，次元が異なる処理であることに注意したい．

大規模図形における衝突検出の大きな問題点は，衝突する相手の図形がどれかを探し出すことである．従来の例えば Boxing（図形に外接する長方形との交叉を調べて，関係ない図形を高速に排除する手法）等による方法では，高速化はするものの結局は総当りであるから $O(N)$ よりオーダを改善することはできなかった．SPF ではこれを局所的処理で $O(N^0)$ にすることができる．**図 16** において衝突点は衝突直前には必ず衝突する相手と同一部分空間の両側で対峙した形になっているから，衝突のチェックは常に運動図形に接する部分空間だけについて行えばよい．

図形Aを移動してゆくと，Aと他の図形との相対位置が変化するのでAの周囲の部分空間が変形してゆくが，ある所まで来ると，部分空間のトポロジカルな位置関係が変化するので，その時点で SPF 上の SPP のつなぎ換え（メンテナンス）を行う．さらに運動を進めてゆくと，やがて図 16 に斜線で示すAの周囲の部

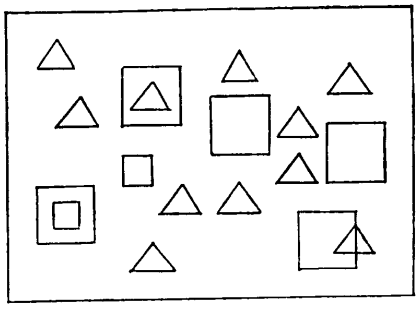


図 15 ⧋ と △△ を探す問題
Fig. 15 Problem to find out ⧋ and △△.

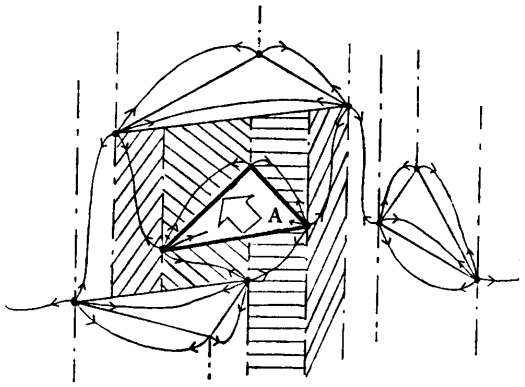


図 16 運動図形の衝突検出，斜線の空間のみ注意すればよい
Fig. 16 Collision detection of moving figure A. It is enough to estimate only hatched sub-spaces.

分空間のどれかで衝突が起こる．これらの部分空間はすべてAの頂点を発する SPP で表現されているから調査判定は容易である．運動図形相互間の衝突も同様に取り扱える．衝突発生または部分空間のトポロジー変化の時点は，運動や図形形状が解析的なものならば部分空間の一辺と対向する頂点の衝突として運動方程式から求めることができる．複雑な形状の場合には，衝突可能性のある（チェックすべき）図形周辺の部分空間の数 n が増加するが，計算時間の増加は $O(n)$ の程度であり，総図形頂点数 N とは例外的な場合（付録 2）を除き関係ない．

衝突の検出は，LSI 設計におけるデザイン・ルールを守ったレイアウト図形のコンパクション，板取り問題におけるつめ合わせ，ロボットの衝突回避問題等広範な用途がある．

4. SPF のメンテナンスと初期生成

前章までに述べた SPF の各種応用と効果は，すべて SPF 構造の図形データベースが事前に作成されて

いることを前提としていた．本章では多角形の入力を例にとって SPF メンテナンスにおける図形1個の追加が，付録2のような例外的な場合を除けば，総図形頂点数 N と無関係で $O(N^0)$ の時間で実行できること．したがってこれの N 回繰返しにより行う SPF の初期生成も $O(N)$ と高速で実用的であることを示す．

SPF への多角形Aの追加は図 17 に示すごとく，最初に A の一つの頂点 a_0 を入力し，次に a_0 を基点として辺を入力し，最近にAの内部空間の SPP に空間属性を付加することで実現する．

最初の点 a_0 の入力は 2.3 節図5～図7に示した手法により点 $a_0(x_{a_0}, y_{a_0})$ を含む部分空間 SPP_{a_0} を探し，図 18(i) のごとく SPP_{a_0} を切断して a_0 を接続する．SPP_{a_0} の検索は道しるべを利用すれば（例外的な場合を除けば）$O(N^0)$ が達成できる．ただし，大量の図形を処理する SPF 初期生成時には，処理時間を少しでも減らすために，道しるべの区間分割数 m（図6）の値の最適化等に十分な配慮をすべきである．

点 a_0 が入力されたら次は辺の入力である．図 18 (ii)～(v) に示すように，a_0 から出発してAの辺に沿って折れ線 $\overline{a_it}$ (i はAの頂点番号) を順に伸ばしてゆくと，$\overline{a_it}$ の先端点 t が部分空間の壁 B_x または B_y を貫通する時点で（これをイベント E_j (j はイベントの発生順につけた番号) と呼ぶ）周囲の空間分割が変化する．辺の入力プログラムはイベントの種類を調べ，その都度対応する SPP のつなぎ換えを行う．起こり得るイベントの種類は上記 B_x, B_y の貫通に $\overline{a_it}$ の始点，折点，終点を加えて表1のようになる．イベントの種類は多くないから，イベントに対応して処理をするプログラムをつくることは容易である．

$\overline{a_it}$ がAの辺を一巡すれば辺の入力は終了する．辺の入力処理は $\overline{a_it}$ の付近の空間に限定されるため，付録2に示すような例外的な場合を除けば $O(N^0)$ の処理である．

辺の入力が終ったら最後に A の内部の SPP に空間属性を付加する（Aの辺のどちら側が図形内部かは，与えられるものとする）．この処理も追加図形 A の内部の部分空間の数（SPP の

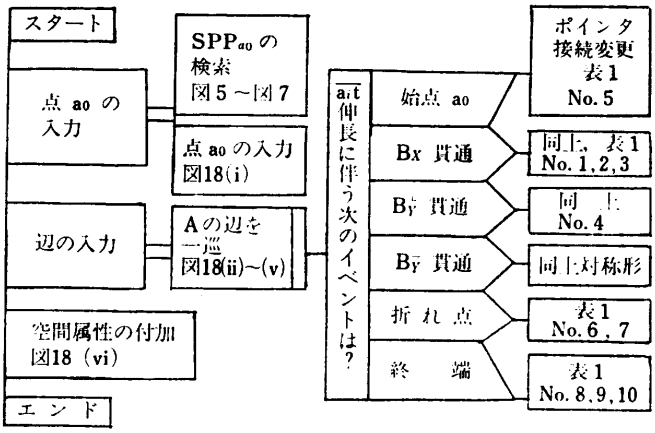
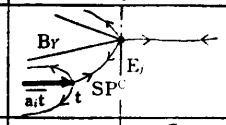
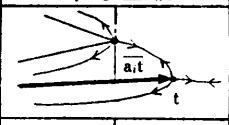
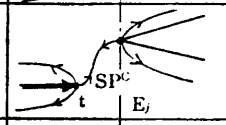
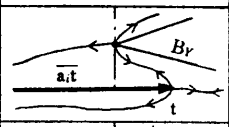
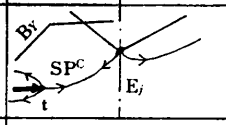
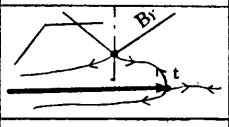
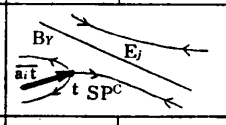
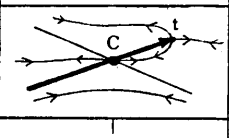
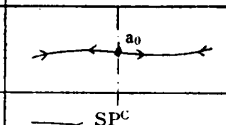
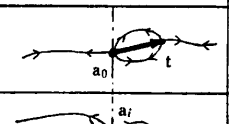
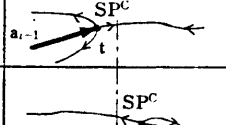
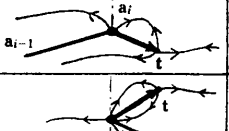
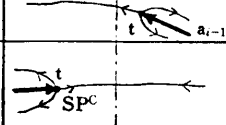
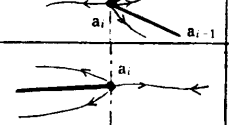
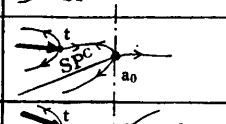
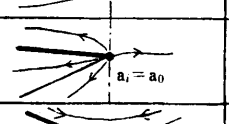
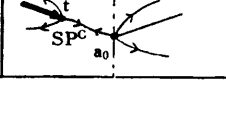
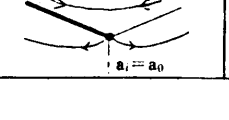


図 17 SPF への図形 A の追加処理 (PAD[9])
Fig. 17 Addition procedure of figure A into SPF (PAD[9]).

表 1 $\overline{a_it}$ 伸長に伴うポインタの接続変更法
Table 1 Maintenance method of pointers along with extension of $\overline{a_it}$.

区分	No.	形状例 Event E_j 発生前	形状例 E_j 発生後	対称形等の存在
Bx 貫通	1			±X ±Y
	2			〃
	3			〃
By 貫通	4			〃
始点	5			±X
折れ点	6			±X ±Y
	7			±X ±Y
終端 線分	8			±X
終端 ポリゴン	9			±X ±Y
終端 ポリゴン	10			±X ±Y

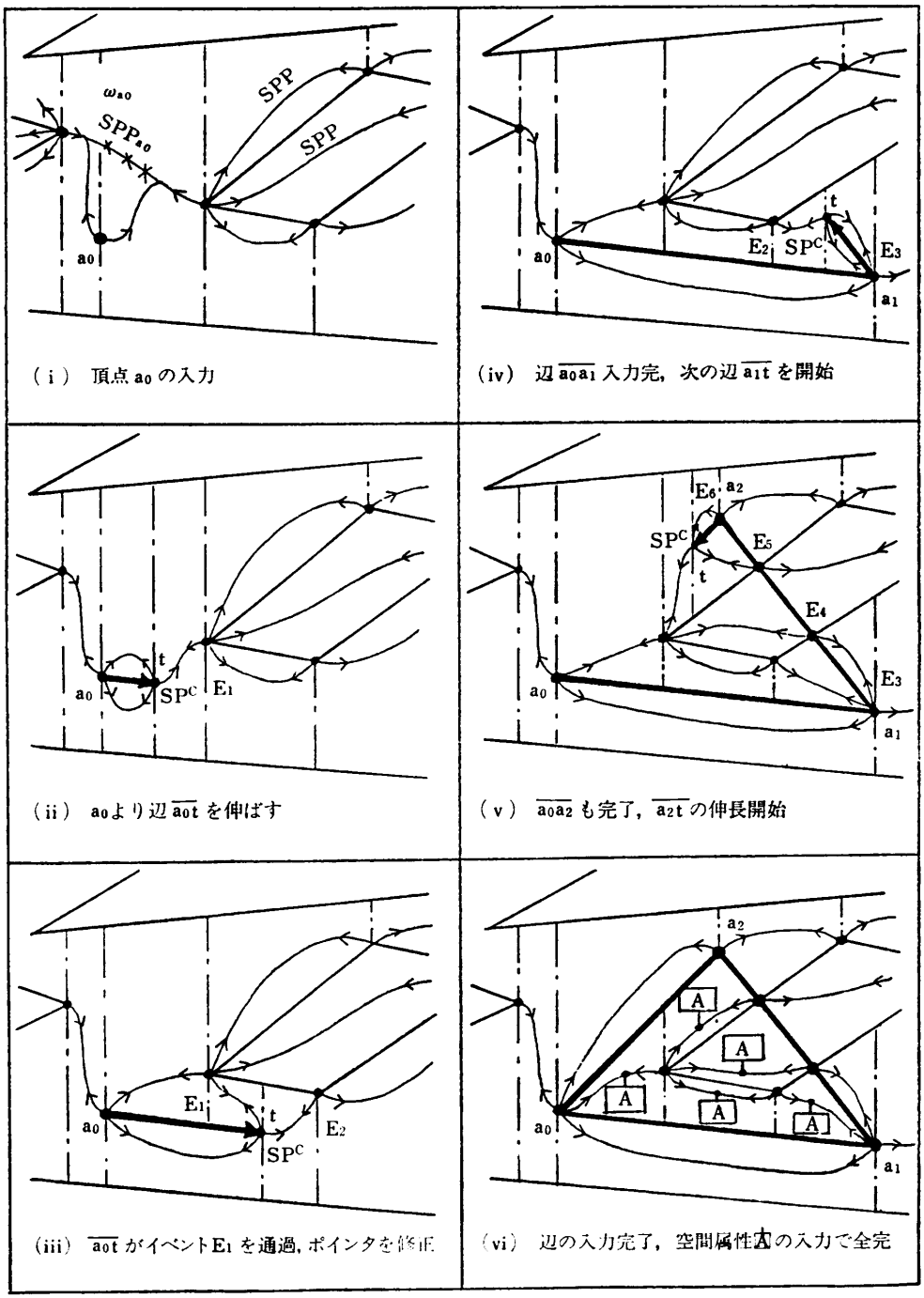


図 18 (i)～(vi) SPF への図形Aの追加手順 (図 17 参照)
Fig. 18 (i)～(vi) Addition procedure of figure A into SPF.

数) が N に関係するような特殊図形を除けば $O(N^0)$ の処理である.

以上の処理のまとめを図 17 に示す. 点の入力, 辺の入力, 空間属性の付加, 各処理とも, 例外的な場合を除けば, $O(N^0)$ の処理であるから, 図形Aの追加全体処理も同じく図形1個当り $O(N^0)$ の処理になる. したがって, これを繰り返して適用する SPF 全体の初期生成時間は $O(N)$ となり, 少なくともオーダの上では, SPF はオンライン用にも, バッチ処理用にも適した図形データベースとなることが期待される.

なお, 図形追加の逆に図形削除があるが, 全くの逆操作なので説明を省略する.

5. むすび

新しい図形空間モデル化方式を提案し，ファイル作成の基本問題と応用の可能性について考察した．本方式の特徴は図形規模によらない局所的図形処理と各種図形，用途に適用できる汎用性にある（付録2に示す例外的な場合を除く）．各種図形処理の基本となる隣接空間，重なり図形の検索が局所的処理で可能なため，これの応用として局所的図形論理操作，運動図形の衝突検出，ファイル・メンテナンス，その他が，例外的な場合を除き $O(N^0)$ の処理時間で実現可能となる．処理時間の長くなる例外的な場合については，LSIのレイアウト図等人為的な図形では発生しやすい可能性もあり注意が必要である．必要があれば付録2で示したような対策を行う．

なお，本方式ではポインタを多用するため，メモリ容量が従来の頂点座標値のみの図形データの数倍になる欠点があるが，これは $O(N)$ であるから最近のハードウェアの進歩を考えれば許容範囲内としてよいと思われる．

本方式については現在試作を計画中である．また，今後の問題として，図形データの階層表現，曲線の具体表現法，等が残されている．

終りに，東京大学工学部杉原厚吉助教授，日立製作所武蔵工場牧本次生工場長，日立製作所ソフト工場越智利夫副技師長，ほか本研究に御助言，御支援下さった方々に感謝の意を表する．

参考文献

1) 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).
2) Lee, D. T.: Computational Geometry—A Survey, *IEEE Trans. on Comput.*, Vol. C-33, No. 12, pp. 1072-1101 (1984).
3) Ousterhout, J. K. et al.: Magic: A VLSI Layout System, *21st Design Automation Conference, Proc. '84, IEEE ACM*, pp. 152-159 (1984).
4) Tsukizoe, 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).
5) 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).
6) 二村良彦ほか：PAD (Problem Analysis Diagram) によるプログラムの設計および作成，情報処理学会論文誌，Vol. 21, No. 4, pp. 259-267 (1980).

付録1 部分空間の表現法

SPP による空間表現は図 A1 のごとく空間表現情報 $\boxed{\omega_i}$ をメモリ節約で省略した形と考えてよい．次に部分空間 ω_i を SPP_i で1対1表現するための一般条件を示す．

(1)図形辺は部分空間の Y 境界 B_Y を形成する．

(2)図形の X^+ 端点（そこから X^+ へ伸びる辺がない）または X^- 端点を通り $\pm Y$ 方向に隣接辺まで伸びる X 境界 B_X を置く．B_X の左右空間の SPP はその端点に接続する．

(3)ω_i の Y 側辺 $B^+_{Y_i}$ と $B^-_{Y_i}$ の交点は SPP_i の接続点となる．

(4)ω_i は $B^+_{Y_i}$, $B^-_{Y_i}$ 上の頂点を通る任意の B_X で分割可能．

以上(1)～(3)は必要条件，(4)は十分条件である．これにより図 A2(i)，(ii)や，本文図13の曲

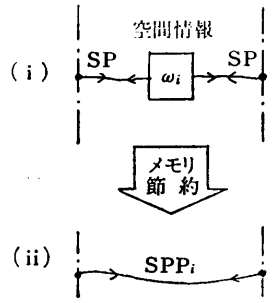


図 A1 空間情報 $\boxed{\omega_i}$ の省略
Fig. A1 Omission of space data $\boxed{\omega_i}$.

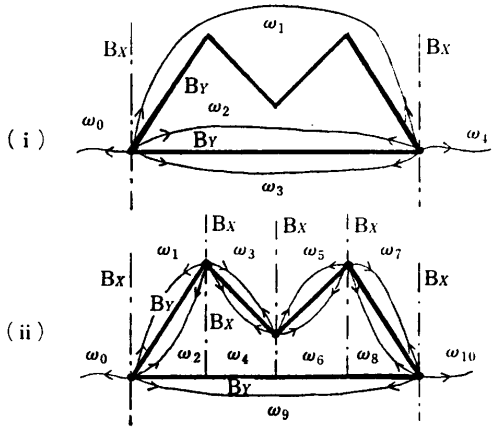


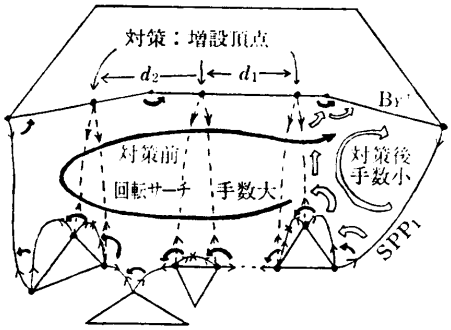
図 A2 部分空間分割方式の例
Fig. A2 Examples of space partition.

線等も許されることになる．なお座標系は斜交，曲線等でもよい．

付録 2 例外的な場合の考察

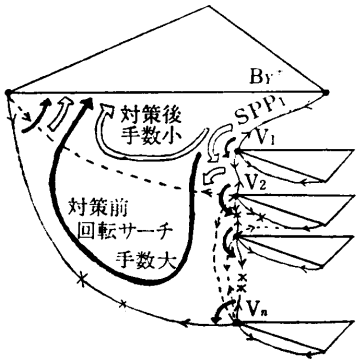
本方式の性能は対象図形の形状，分布等に影響され，図 A3，図 A4 等例外的な場合では回転サーチの手数が増加して性能が低下する．対策としてはデータベース生成時に回転サーチの手数を一定値以下に抑えるような間隔で，図A3 に示すような図形形状に影響を与えない頂点を増設する方法がある（その代り副作用として頂点が増加する）．なお図A4では頂点の X 座標値が等しいので，上記対策の極限として X 値の等しい点に微小間隔だけ離れた複数個の増設頂点を設置してもよいが，本文図3(i)で示した X 値の等しい頂点はどちらかが右にあると見なす方法を適用して，各頂点の X 方向並び順を変えることにより図示のように改善する方法もある．この場合各枝の深さが等しいバイナリー・トリー状に並べれば，回転サーチの手数は $O(\log n)$，(n は並んだ頂点の数) 程度となり，増設頂点も不要で好ましい（プログラムが複雑化するので厳密ではないが簡単な手法による方法もある）．

以上例外的な場合と対策の例を述べたが，対策はいずれも副作用があり，トレード・オフが必要となる．



対策：回転サーチの手数が一定値以下になるような間隔 ($d_1, d_2, \cdots$) で上側の辺に頂点を増設

図 A 3 例外的な場合と対策
Fig. A 3 Special case and its counter measure.



対策：$V_1 \sim V_n$ の X 座標値並び順変更

図 A 4 例外的な場合と別の対策法
Fig. A 4 Special case and its another counter measure.

また上記以外に N 個の図形が同一形状で同一個所に重なった図形や，無限の頂点を持つフラクタル等本方式に適さない図形もあり得る．しかし，LSI マスク・パタン等実用的な対象では図形サイズ，分布等も比較的均一で，1図形当りの頂点数も無闇に多くはならない場合が多く（文献5)），また大規模図形では本文 2.5 節図 14 に示すページ分割を行うことも一種の対策になるので，実用上あまり大きな問題にはならぬと思われる．

(昭和 60 年 5 月 16 日受付)
(昭和 61 年 9 月 10 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，主としてプロセス入出力装置，磁気ドラム，CRT，電源装置等アナログ関係設計を担当．昭和 50 年より日立武蔵工場で半導体 CAD 等を担当，現在に至る．副技師長．電子通信学会会員．