計算誤差による暴走のない図形演算アルゴリズム：スペースモデルによる実現†

大沢 晃††

計算誤差により図形演算プログラムが暴走する問題は，従来から図形処理における大きな未解決課題とされていた．これに対し，従来普通には「非常に接近した図形要素は同一点を占めている」とみなす対策が行われていた．しかしこれはある場合にはかえって暴走の原因となり，対策として不完全であった．本論文ではスペースモデルにより図形のトポロジーを表現し，「どんなに接近した図形要素も，たとえ両者の座標値が等しくなっても，トポロジカルには同一点とは考えない」手法を採り，トポロジーに矛盾が起きないようにデータ構造を生成する，新しい対策法と実験結果につき述べる．座標値よりトポロジーを優先したこと，図形の正しさは追及せず，トポロジーが矛盾しないことを条件としたこと，スペースモデルを使うことで起こり得る形状の種類が限定でき，すべての場合の列挙が可能になったこと，などが本方式のキーポイントである．スペースモデルによれば，二次元・三次元，注目，局所集合演算，運動図形の衝突検出，等，多彩な機能が平均的に $O(1)$ の対話速度で実現できるが，本方式により本質的に高信頼度が得られ，実用化の基礎を固めることができたと考えられる．

1. はじめに

計算誤差により図形演算プログラムが暴走する問題は，図形処理における大きな未解決課題とされていた．例えば図形の辺，頂点，交点等の図形要素が非常に近接しているときに交叉計算を行うと，計算誤差のため図形の交叉の有無や並び順の判定を誤り，図形のトポロジーが矛盾を起こすことがある．この種の矛盾を含むデータにプログラムがさらに処理を続けると暴走を起こし，無限ループやデータ破壊などの致命的事故にいたる．これに対し従来「非常に接近した図形要素は，同じ位置を占めている」とみなす方法[8)~10)]，不定長整数を分母・分子とする有理数で演算結果を表現する方法[5)]，演算誤差を毎回見積りその誤差が致命傷にならぬような前処理を演算のたび施す方法[6)]，など各種の対策[7)]が提案されているが，いずれも対策として完全ではなかった．また桁数を必要精度まで増やして誤差を全く発生させない方式もある[3)]が，精度保持のため5倍長のデータを要する問題がある．その他，最近ボロノイ図を使う方法[4)]が提案されているが，現状では点図形のみに限定されている．

従来暴走対策が困難であった理由は，暴走がトポロジーの矛盾に起因する問題であるにもかかわらず，その表現法が未確立のため，的確な矛盾回避処置ができなかったためと思われる．本論文ではスペースモデル[1),2)]のトポロジー表現力を活用し，座標値よりトポロジーを優先させ，計算誤差があっても矛盾のないデータ構造を生成して暴走を防ぐ方式を提案する．なおこれは矛盾をなくすだけで，スペースモデルの原理を変更するものではないから，そのデータ構造の上で，注目，局所図形集合演算，運動図形の衝突検出，その他を平均的に $O(1)$ の速度で実現する機能性能には変化はない．

以下第2章では，従来の問題点を明確にしてスペースモデルによる対策の基礎を作り，第3章で対策内容につき述べる．また第4章では，各種データに対して暴走が起きないことを確認した実験結果を示す．なお付録の，二次元スペースモデルの原理を参照されたい．

2. 従来の問題点分析と対策の方向付け

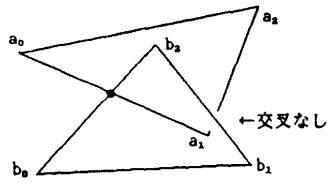
2.1 暴走の原因：トポロジーの矛盾の発生

従来のアルゴリズムは，例えば多角形の OR 演算で交叉の形状を**図1**（i）のごとく誤認して，トポロジーの矛盾から暴走する場合がある．この原因は計算誤差により，交叉の判定を誤るからである．これに対し従来普通には「非常に近接した図形要素は同一位置を占める」とみなす対策が行われてきた．これによれば，近接した辺と頂点は強制的に結合されてしまうので，この種の矛盾は起きない．しかし，**図2**（i）の場合に同じ方式を適用すると，同図（ii）のように，二交点 c_1 と c_2 は近接していて同一点と認識されるが，頂点 a_1 が離れている場合には，二辺が $c_1 \equiv c_2$ 点で

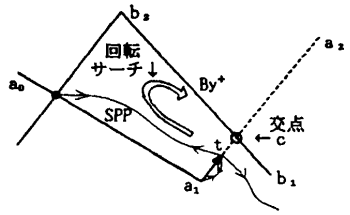
† Failure Free Geometrical Algorithm against Calculation Error: Realization by Means of Space Model by AKIRA OHSAWA (Keihin Technical College, Hitachi Ltd.).

†† 日立製作所京浜工業専門学院

いったん交叉し，そのほかに頂点 a_1 でもう一度交叉するという，トポロジカルに矛盾したデータとなって暴走を起こし，対策として不完全であった．

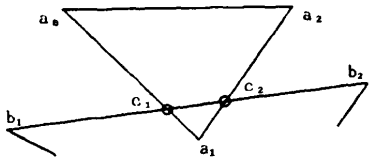


(i) トポロジーの矛盾 ⇒ 暴走

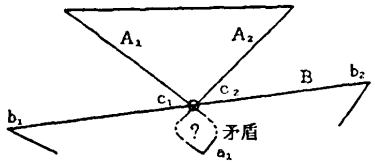


(ii)SPFによる正しい交叉判定
(回転サーチによりBy⁺を検索)

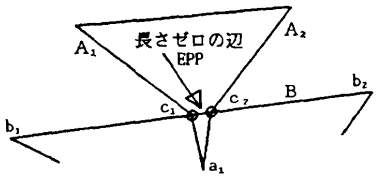
図 1 計算誤差による矛盾と SPF による対策
Fig. 1 Inconsistency caused by calculation error and preventive measure using SPF.



(i) 正しい図形 ⇒ 暴走せず



(ii) c_1, c_2 を同一点視(従来の対策)⇒暴走



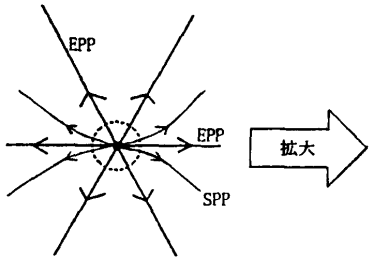
c_1c_2 は同一座標でも別々とする⇒暴走せず
(iii)今回のスペースモデルによる対策

図 2 近接点を同一点とみなす矛盾と対策
Fig. 2 Measure to prevent inconsistency.

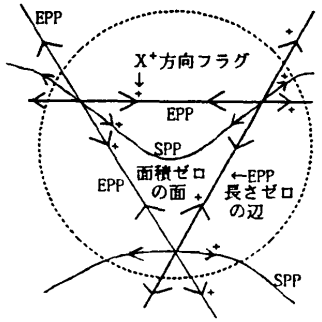
2.2 スペースモデルによる暴走対策の考え方

数値計算で計算誤差による交叉の誤判定をなくすことは難しい．しかし図1(i)で辺 a_0a_1 を入力した時点で，a_1 の位置が b_1b_2 の下側にあると判定され，かつ a_2 が b_1b_2 より上方にある場合には，数値計算の結果にかかわらず強制的に交点を設置することにすれば，トポロジーの矛盾をなくすことができる．これを実現するには，頂点 a_1 が与えられたときに，a_1 の上側に隣接して辺 b_1b_2 が存在することが効率良く検索できなければならない．ここではそのための仕掛けとして，図形をスペースモデルのデータ構造 SPF (Space Pointer File：付録参照) で表現する．図1(ii)には，a_1 の入力の次に短い辺 a_1t の入力が始まる場合の，t 点の付近の SPF を示す．これによれば，入力辺の先端 t から白抜き矢印で示す回転サーチ(付録参照)を実行して，t 点の上側の辺 b_1b_2 を求めることができる．したがって，もし入力の次の頂点 a_2 の位置が b_1b_2 より十分に上側にあれば，交点計算をしなくても（したがって誤差に関係なく）交叉ありと判定できる．

また図2(i)，(ii)に関しては，本方式では従来の「非常に接近した図形要素は同一位置を占めている」とみなす対策は行わず，逆にどんなに接近した図形要素も，たとえ両者の座標値が等しくても，別個のものは別個と認識してトポロジーを保存し，問題を回避する．すなわち，もし c_1 と c_2 の距離が近くて数値表現の最小分解能以下である場合には，両者の座標値は等しくなるが，トポロジカルには離れているとみなす．このため，同図(iii)に示すように，c_1 と c_2 とは別々の交点とし，両者の間に長さゼロの辺を表現するス



(i)多重交点



(ii)長さゼロのEPPで単純交点に分解，
X⁺方向フラグによる有向グラフ表現

図 3 長さゼロの辺による多重重なり点の表現
Fig. 3 Representation of multiple cross point by use of zero length edge.

ペースモデルのポインタの組 EPP (Edge Pointer Pair：付録参照) を設置する．これで近傍を同一点と誤認することによる暴走は防止される．なお座標値が等しい場合や，さらに計算誤差で座標値の大小が本来の順序と逆転した場合でも，正しいトポロジー（空間の隣接関係）を保持できる必要がある．そのためポインタの組 SPP (Space Pointer Pair) および EPP に X^+ 方向（X座標値が増加する方向，ただしXの増分がゼロの場合にはYが増加する方向：言い替えればX，Yを辞書順に見た値が増加する方向）の情報をフラグとして持たせる．**図3**は複雑な交点が同一座標点に重なる多重交点を，長さゼロの辺を使って単純な二辺の交点に分解し，+印のフラグで X^+ 方向の順序を表現した例である．この方法で，誤差を含む多数の辺や頂点，交点が重なる場合のトポロジー判定の混乱を防止することができる．SPF では空間のトポロジーをポインタで表現し，座標値は頂点や交点に付属する一種の属性値とみなしているとも言える．これにより地図や加工物の誤差を含む測定値の扱いも可能になる．

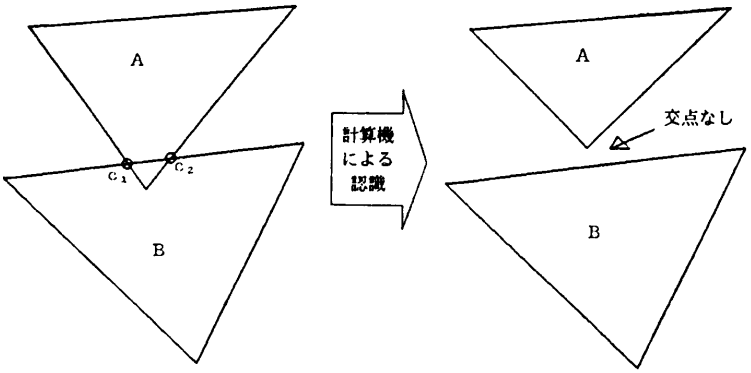


図4 座標値もトポロジーも正しくないが矛盾はなく許容すべき例
Fig. 4 Not correct but consistent example on both topology and coordinate values.

2.3 対策の必要十分条件：矛盾のない SPF 生成

SPF では図形辺の交点も頂点の一種として陽に表現する．このため交叉計算は SPF 生成時点で行う必要があり，計算誤差の対策もその時点で行う．対策が完全なら生成される SPF は，図1のような矛盾は含まないクリーンなものになり，図形演算は暴走なく行われる．例えば図形の OR は SPP の持つ属性が空でない空間を列挙し，AND は属性の重なり部分のみを取り出せばよい（付録）．これらの演算は SPF のポインタで表現されたトポロジー情報を検索処理するが，交点計算は行わないから，この時点で新たな計算誤差を発生することはない．したがって暴走対策は SPF 生成時点で完結する．言い替えれば，矛盾のない SPF 上では暴走しないように図形演算プログラムを作成することにすれば，トポロジーに矛盾のない SPF を生成することが，暴走対策の必要十分条件になる．なお**図4**のように，本来交叉している図形を計算誤差により交叉していないと判断する場合や，その逆の場合がある．これは座標値の数値もトポロジーも正しくないが，入力図形の位置が少しずれていた場合と同じで，トポロジーに矛盾はなく暴走は起きず，許容される．必要なことはトポロジーの矛盾を生じないことであり，図形の正しさを追及することではない．

3. スペースモデルによる暴走対策

3.1 対策の準備

3.1.1 矛盾発生個所の特定

スペースモデルによる暴走対策は，前述のごとく矛盾のない SPF を生成することで実現される．本項では生成手順を分析して，矛盾発生個所を特定する．

SPF の生成は，すでに r 個の多角形による SPF が生成済みであるとして，これに第 $r+1$ 番目の多角形を追加入力することの繰返しによって行う[1]．したがって SPF への1個の多角形の入力が，形状にかかわらず常に矛盾なく行われれば，暴走対策は完成する．入力多角形は頂点座標列 $A\{a_0(x_0, y_0), a_1(x_1, y_1), \cdots, a_i(x_i, y_i), \cdots\}$ で与えられる．これを SPF に追加する手順を**図5**および**図6**の PAD で示す．矛盾の原因となる計算誤差は，演算の途中で丸め処理（四捨五入，切り捨て）があると発生する．丸め処理は図6の中の辺の交叉計算（★2）および点と辺の位置関係の判定計算（点が辺のどちら側にあるか：★1，★3）の中の割算で起こる．以下 ★1〜★3 について検討する．

★1. 入力の始点 a_0 を含む空間 $SPPa_0$ を探すときに点と辺の位置関係の判定（付録）で計算誤差を発生し，始点と終点の空間が違うために図6の★3で多角形が閉じない問題を起こす．対策は 3.5 節で述べる．

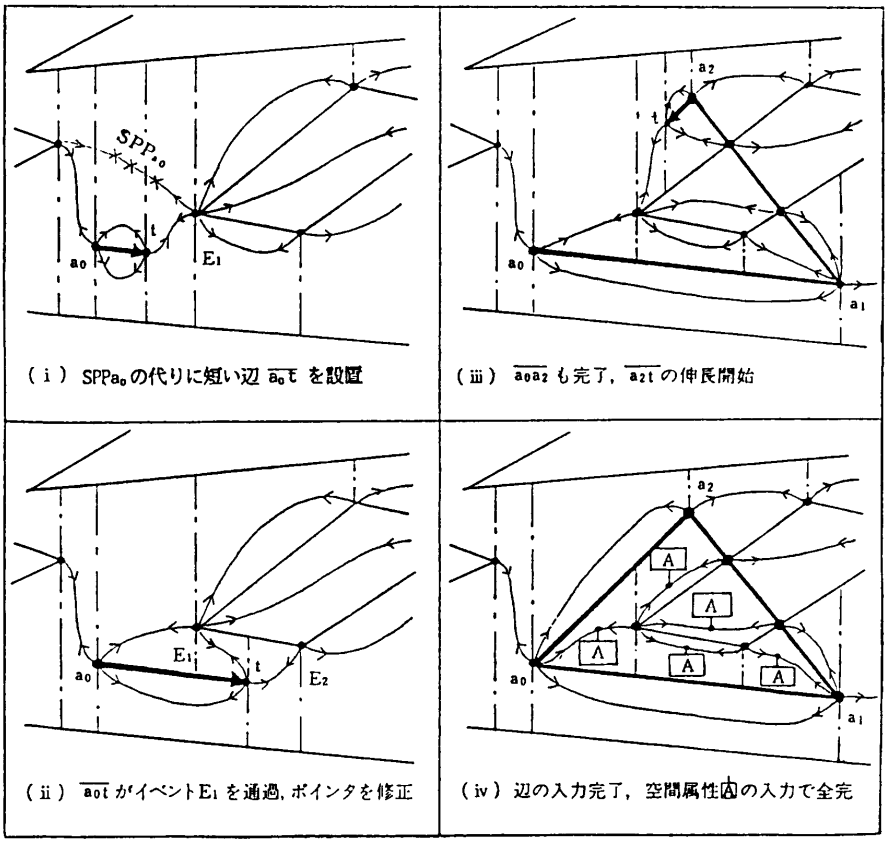


図 5 図形Aの入力処理
Fig. 5 Input process of figure A.

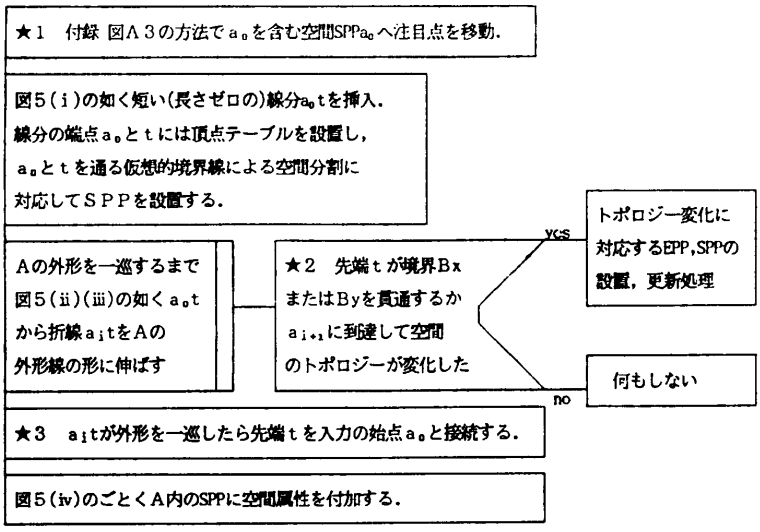


図 6 SPF への多角形の追加手順 (PAD)
Fig. 6 Addition procedure of a polygon into SPF.

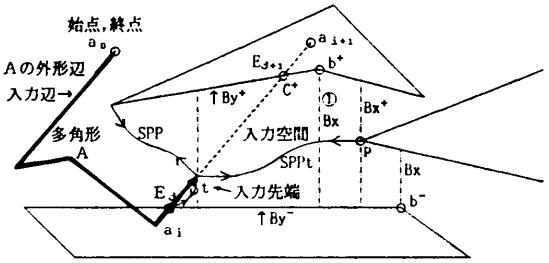


図 7 次のイベント E_{j+1} は入力空間の外周で発生
Fig. 7 Next event E_{j+1} happens on the border of input subspace.

★2. 交点の計算誤差でイベント* すなわちトポロジーの変化の検出を誤って，EPP, SPP の設置，更新を行うと，図5のデータ構造に矛盾（ポインタの接続誤り）を作り込み，暴走を起こす．

ここで**図7**により入力 a_it がイベント E_j を過ぎて t 点まで伸びてきたときに，次のイベント E_{j+1} を求める手法と，矛盾の原因となる E_{j+1} の検出誤りについて説明する．a_it は X^+ 方向に伸びて，SPPt で表現されている入力空間の外形を構成する三つの境界線 By^+, By^-, Bx^+ のどれかと交叉するか，またはAの次の頂点 a_{i+1} に到達する．これらを次のイベント E_{j+1} の候補点という．最終的な E_{j+1} は，上記候補点のうち，a_it の伸長に伴い最初に発生する点（この場合はX座標値が最も小さい点）として求められる．前三者の候補点は，t 点から回転サーチ（付録参照）を含む SPF のポインタ検索で求めた By^+, By^-, Bx^+ と入力辺との交叉計算で求められる．もしここで計算誤差があると，X座標値の判定を誤り，誤った候補点を E_{j+1} と誤認することがある．計算誤差が避けられない以上，常に正しい E_{j+1} を求めることは望めない．しかし，正しくなくとも，トポロジーに矛盾がないように E_{j+1} を決定することはできる．その方法を次節以下で述べる．なお E_{j+1} の候補点のうち，頂点 a_{i+1} の座標値は入力データで与えられるもので，計算誤差は含まない．

★3. 上記 ★1 の結果，a_it の先端 t が A の外形を一巡した終点で a_0 と接続できなくて暴走にいたる場合がある．本件は3.5節で述べる．

3.1.2 SPF による交叉計算対象の限定

SPF の利点は，空間の分割とポインタによる構造化で，入力との交叉計算をするべき相手の辺を，全図形のうちから By^+, By^-, Bx^+ の三つに特定できるため，矛盾防止のために検討すべき形状の種類を，上記の三つの境界線と入力辺の形状の組合せの数だけに限定できることである．これにより，すべての場合を列挙して，矛盾を生じないような E_{j+1} を決定することが可能になる．なお，相手の限定により無駄な交叉計算がなくなり，処理も高速化する．

3.1.3 凸空間の取扱いの簡素化

付録に示すように，SPF では頂点周りの空間を挟む二辺のなす角度が 180° 以上（凹空間）のときのみ，その空間を仮想線 Bx で部分空間に分割した．このため本来は図7の頂点 b^+ の直下①の位置には Bx は存在せず，a_it の伸びていく入力空間は p 点まで広がり，その Y^+ 側境界線 By^+ は b^+ 点で曲がる折れ線となる．しかし入力 a_ia_{i+1} が b^+ の近傍で By^+ と交叉する場合には，計算誤差のため b^+ の左右の辺と重複交叉したり，または逆に両辺ともに交叉しないと判定して矛盾を起こすことがある．この対策を簡単にする一つの方法は，b^+ には SPP は接続しないが，同図①の境界線 Bx を想定して，a_it の伸びていく入力空間が b^+ 点までで終わり，外形辺 By^+ が常に直線になるように空間を分割することである．なお，入力が By^+ と同時に Bx と重複して交叉する場合には，By^+ との交叉を優先して矛盾を避ける．

3.2 次のイベント E_{j+1} の候補点の求め方

本節では E_{j+1} の候補点を求め，次に3.3節で，その中から矛盾を生じないように E_{j+1} を決定する．

3.2.1 トポロジーに矛盾を生じない交叉判定法

まず E_{j+1} の候補点のうち，入力と By^+ との交点 C^+ を，トポロジーに矛盾を生じない形で求める．図1(i)の問題防止のため，イベント E_j（図の a_1）通過後の，By^+ との交叉の判定は「t 点から回転サーチで求めた By^+ は，t 点より上側にあるはず」というトポロジカルな知識を使い，**図8**のプログラムで行う．この方式では，例えば過去に a_1 を入力した時点で，計算誤差のため本来は a_1 より下側にあるはずの辺が上側にあると誤認されて By^+ と名付けられていたとすると，SPF のポインタがそのようにセットされているので，t 点からの回転サーチでも同じ By^+ が上側の辺として検索される．したがって，a_2 が

* a_it を入力多角形の辺に沿って伸ばすとき，空間のトポロジー（隣接関係）は変化せず，空間の形だけがアナログ的に変化するときは，SPF のデータ構造の変更は必要はない．しかし，先端 t が空間の境界 By^+, By^-, または Bx^+ を貫通するか，または，a_{i+1} に達した時点でトポロジーが急変するから，これを反映して EPP, SPP の設置，更新を行う．このトポロジー変化をイベント E_j（j はイベント番号）と呼ぶ．処理はイベント発生時のみに行うため，a_it は連続でなく飛び飛びに伸ばせばよい．

By^+ より上方に在るときには，交点計算とは無関係に交点が設置される．これで a_0a_1 を入力した経過と，a_1a_2 を入力する場合との矛盾はなくなる．なお交点の座標値は誤差を含むが暴走には結び付かない．

交叉判定は**図9**（i）～（iv）に示す入力と By^+ のすべての形態について行われる．なお図8のプログラムは，図9（i）で a_{i+1} が By^+ に極めて近い場合には，本来は a_{i+1} が By^+ より上側にあるのに（ii）のように下側と判断し，E_{j+1} は交点でなく a_{i+1} であると誤認することがある．しかし，これは図4で入力図形の位置が少しずれた場合と等価で矛盾はなく，暴走は起きない．このため次のイベントを a_{i+1} と判定した場合には，判定をそのまま採用する．また辺 a_it が b^+ の近傍を通るときにも，図9（iii）と（iv）の判定を誤ることがあるが，同様に暴走は起きない．By^- 側も同様である．

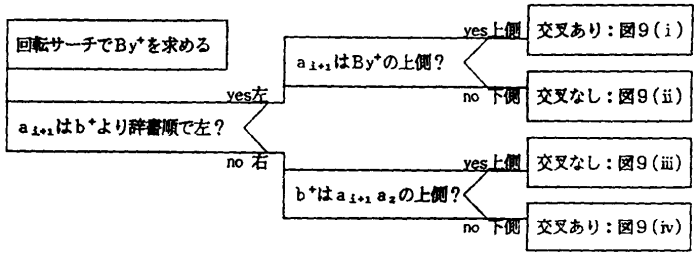


図8 入力と By^+ との交叉を求めるプログラムの PAD（図9参照）
Fig. 8 To find out crossing between input and By^+ (ref. Fig. 9).

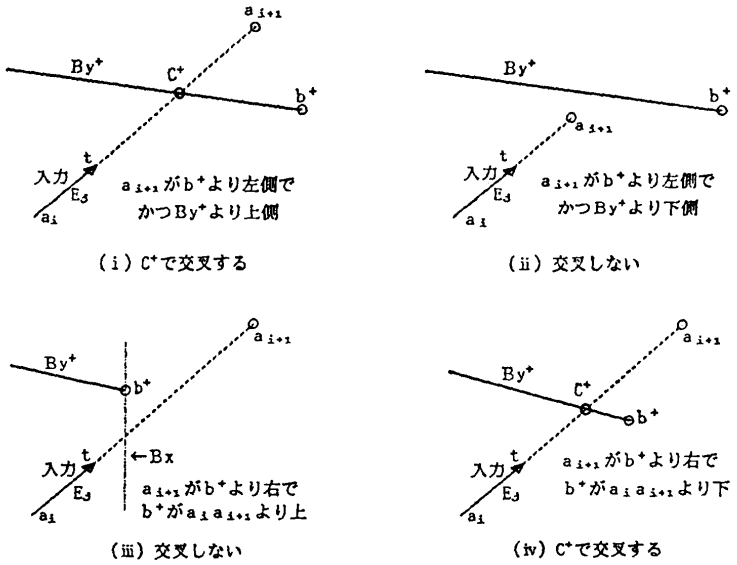


図9 入力と By^+ との4種類の交叉を求める（図8参照）
Fig. 9 To find out four types of crossing between input and By^+ (ref. Fig. 8).

3.2.2 必ず結果を出す交点計算

前項で交叉ありと判定されたときには交点の座標値が必要になるが，二線分の交叉計算では交点が求まらないことがある．これを避けるために線分を延長した二直線の交叉計算で交点を求める．これなら平行でない限り必ず交点が求まる．交点が求まれば計算誤差があってもトポロジーに矛盾は起きない．平行線の場合には計算式の分母がゼロになるので，ゼロを検出し，平行部分の（例えば）中点値を交点の座標値とする．なお，暴走には結び付かないが，求めた交点が元の辺の x，y 座標値の最大／最小値の範囲外となるのは不合理なので，内側の辺の端点座標値に置き換えて精度の向上を図るのがよい．

3.2.3 Bx との交点および a_{i+1} のX座標値

E_{j+1} の候補点のうち，入力と Bx との交点のX座標値は Bx を形成する頂点のX座標値に等しい．これには By^+ の先端 b^+，By^- の先端 b^-，および SPPt の先端 p，の三つがある．前節で述べたように，E_{j+1} を決めるために必要なのはX座標値だけであるから，以下便宜上これら Bx との交点の代わりに，それぞれ b^+，b^-，p を用いる．これらは t 点からの回転サーチを含むポインタ検索で簡単に求められる．また a_{i+1} の座標値は入力データで与えられる．以上で E_{j+1} の候補点 $(C^+, C^-, b^+, b^-, p, a_{i+1})$ のX座標値はすべて判明した．このうちから矛盾を避けて E_{j+1} を決定する方法を次節に示す．

3.3 候補点の中から E_{j+1} を決定する

3.3.1 トポロジカルな制約条件

E_{j+1} を求めるために，前節の候補点を引き数とし，そのうちのX座標値が t 点に最も近いものを E_{j+1} として返す関数 nextev $(C^+, C^-, b^+, b^-, p, a_{i+1})$ を作る．注意すべきは，候補点の空間的な並び順にはトポロジカルな制約があることである．入力空間の形状により制約条件が違うので，**表 1～3** を含むすべての場合を網羅した「イベント E_{j+1} 判定表」を作成して，制約条件をチェックする．例えば表1の中の図③では，

表 1 イベント E_{j+1} 判定表（その1）
(SPPt の先端) p点が左に凸
Table 1 Event E_{j+1} find out table (No. 1).

		(1) By^+は$a_i a_{i+1}$と交叉する	(2) By^+は 入力辺$a_i a_{i+1}$と交叉せず	
			b^+にSPPが接続しない	b^+にSPPが接続する
By^-は$a_i a_{i+1}$と交叉する		図① E_{j+1}=nextev(C^+, C^-, p, a_{i+1})	図② E_{j+1}=nextev(b^+, C^-, p, a_{i+1})	図③ E_{j+1}=nextev(C^-, p, a_{i+1})
By^-は$a_i a_{i+1}$と	b^-にSPPが接続しない	図④ E_{j+1}=nextev(C^+, b^-, p, a_{i+1})	図⑤ E_{j+1}=nextev(b^+, b^-, p, a_{i+1})	図⑥ E_{j+1}=nextev(b^-, p, a_{i+1})
交叉せず	b^-にSPPが接続する	図⑦ E_{j+1}=nextev(C^+, p, a_{i+1})	図⑧ E_{j+1}=nextev(b^+, p, a_{i+1})	図⑨ E_{j+1}=nextev(p, a_{i+1})

表 2 イベント E_{j+1} 判定表（その2）
(SPPt の先端) pが右に凸
Table 2 Event E_{j+1} find out table (No. 2).

		(1) By^+は$a_i a_{i+1}$と交叉する	(2) By^+は入力$a_i a_{i+1}$と交叉せず	
			b^+にSPPが接続しない	b^+にSPPが接続する
By^-は$a_i a_{i+1}$と交叉		図① E_{j+1}=nextev(C^+, C^-, a_{i+1})	図② E_{j+1}=nextev(b^+, C^-, a_{i+1})	図③ E_{j+1}=nextev(C^-, a_{i+1})
By^-は$a_i a_{i+1}$と	b^-にSPPが接続しない	図④ E_{j+1}=nextev(C^+, b^-, a_{i+1})	図⑤ E_{j+1}=nextev(b^+, b^-, a_{i+1})	図⑥ E_{j+1}=nextev(b^-, a_{i+1})
交叉せず	b^-にSPPが接続する	図⑦ E_{j+1}=nextev(C^+, a_{i+1})	図⑧ E_{j+1}=nextev(b^+, a_{i+1})	図⑨ (注) 強制的にC^+(又は C^-)を設置 E_{j+1}=nextev(C^+, a_{i+1})

(注) 入力辺がBy^+, By^-側辺共に交叉せず，かつ$E_{j+1} \neq a_{i+1}$は矛盾：対策として強制的に交点 C^+(または C^-)を設置。座標値はpに等しくする。

表 3 イベント E_{j+1} 判定表（その3）
(SPPt の先端) p点が下に凸
Table 3 Event E_{j+1} find out table (No. 3).

		(1) By^+は$a_i a_{i+1}$と交叉する	(2) By^+は入力辺$a_i a_{i+1}$と交叉せず	
			b^+にSPPが接続しない	b^+にSPPが接続する
By^-は$a_i a_{i+1}$と交叉する		図① E_{j+1}=nextev(C^+, a_{i+1}) SPPtがBy^+に接続するのでC^+は必ずC^-よりtに近い	図② E_{j+1}=nextev(b^+, C^-, a_{i+1})	図③ E_{j+1}=nextev(b^+, C^-, a_{i+1})
By^-は$a_i a_{i+1}$と交叉せず	b^-にSPPが接続しない	図④ E_{j+1}=nextev(C^+, b^-, a_{i+1})	図⑤ E_{j+1}=nextev(b^+, b^-, a_{i+1})	図⑥ E_{j+1}=nextev(b^+, b^-, a_{i+1})
	b^-にSPPが接続する	図⑦ E_{j+1}=nextev(C^+, a_{i+1})	図⑧ E_{j+1}=nextev(b^+, a_{i+1})	図⑨ E_{j+1}=nextev(b^+, a_{i+1})

SPPt 以外の SPP が接続している b^+ は必ずp点より右側でなければならぬという制約がある．しかしX座標値の比較では，誤差で b^+ が左と判定して矛盾を起こすことがあるから，この場合の b^+ のように座標値を比較すべきでない候補点は，nextev ( ) の引き数から除外しておく（表参照）．また例えば表3図②では，b^+ より右側のはずの p 点が左側と判断されると矛盾を起こすので，pは引き数から除外してある．

この方式は言い替えれば，並び順を誤って判断しても矛盾を起こさない候補点だけを引き数とすることである．例えば表1の図②では引き数 b^+ とpのX座標値の順序が誤っても SPF の構造（ポインタの接続）は変化せず矛盾は起きない．また，表のすべての項目に引き数 a_{i+1} が入っているが，これは a_{i+1} は判定を誤っても図4の理由でトポロジーの矛盾が起きないからである．なお特別の場合として，表2図⑨で a_{i+1} がずっと遠方の場合，入力が By^+, By^- のどちらとも交叉しないはずはないので，強制的に C^+ または C^- を設置する．

本方式によれば，誤差の大きさによらずトポロジーに矛盾は発生しないから暴走は起きない．図形の精度が悪くなって使いものにならなくなるのが，このシステムの限界になる．以上の検討結果をいくつかの例についてチェックした．**図 10** はその一部である．

3.3.2 イベント E_{j+1} 判定表の構成

表1～3の例を含む「イベント E_{j+1} 判定表」を作成し，前項のトポロジカルな制約により nextev ( ) の引き数を決定した．以下，対策の完全性を示すために，これらの表ですべての形状が列挙できることを説明する．

$a_i t$ が X^- 向きの場合は X^+ 向きの場合の対称形なので説明を略す．X^+ 向きのすべての形状を網羅するには，E_{j+1} の各候補点 (C^+, C^-, b^+, b^-, p, a_{i+1}) のあらゆる組合せを表にする必要がある．ただし a_{i+1} は図4の理由で誤っても暴走しないので除外する．

次に，SPPt の先端の頂点pの形状は下記のように4通りなので四つの表に分けた．

pが左に凸：表1，pが右に凸：表2，

pが下に凸：表3，pが上に凸：表4

（表4は表1とX軸に関する対称形のため掲載省略）

残りの (C^+, C^-, b^+, b^-) の形状は表1～4の中で分類する．そのため各表とも横軸に C^+ と b^+ を，縦軸に C^- と b^- を取り，各軸にそれぞれすべての状態を並べて一覧表にする．まず横軸は，C^+ の取り得る

状態，すなわち(1) a_ia_{i+1} と By^+ が交叉する場合と(2)交叉しない場合の二つに分類する．次に(2)を，b^+ に SPP が接続しない場合と接続する場合の2種類に細分した．(2)をこのように分類する理由は，前項で表1の図②と図③の例で説明したように，前者では b^+ を引き数に入れる，後者では入れない，ように差があるからである．なお横軸の(1)は b^+ で細分する必要はない．これは C^+ 点で交叉するので，b^+ は t 点から見て C^+ 点より遠方のはずというトポロジカルな制限条件があり，したがって b^+ は E_{j+1} になり得ず，判定に影響しないからである．当然，各表の横軸(1)では nextev ( ) の引き数には b^+ を入れていない．

表の縦軸についても C^- と b^- に関して同様に分類する．これですべての候補点の状態の組合せが表にできたので，起こり得るすべての場合が尽くされる．なお，By^+ や By^- が鉛直辺の場合も，少し傾斜していると考え，ここに含めている．

X^- 方向についても同様に四つの表ができる．

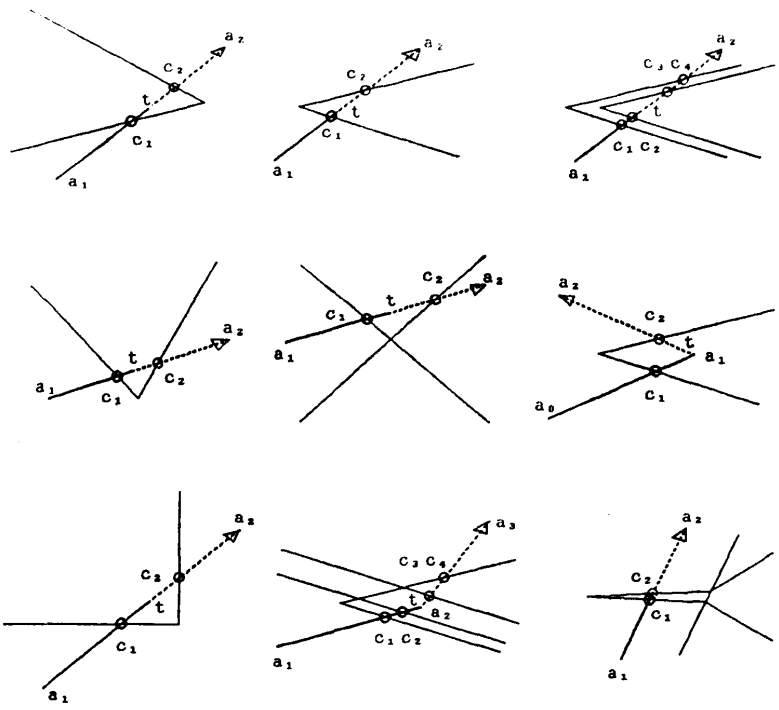
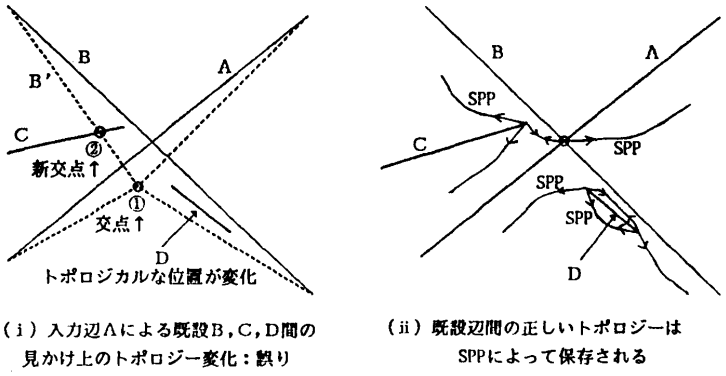


図 10 計算誤差問題をチェックした例
Fig. 10 Some examples of checking calculation error problems



(i) 入力辺Aによる既設B，C，D間の見かけ上のトポロジー変化：誤り

(ii) 既設辺間の正しいトポロジーはSPPによって保存される

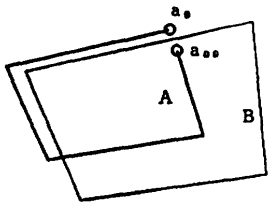
図 11 誤差の影響の伝播防止
Fig. 11 Prevention of error effects propagation.

3.4 誤差影響の伝播防止

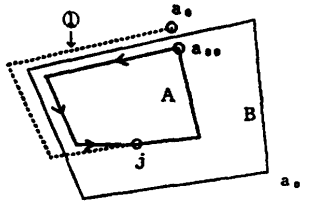
図 11(i)で入力辺Aが既設辺Bと交叉し，誤差により①に交点ができるとBが B′ に変わる．ここでもし B′ がCと交叉して新交点②を設けると，Cが C′ に動いてさらに…と影響が伝播し厄介になる．しかし B→B′ は見かけ上であり，本来Bの位置が動くはずはないので，交点②を設けなければ問題は起きない．Dも同様である．トポロジーは同図(ii)のように SPP で保持される．要するに何もしなければよく簡単である．

3.5 入力多角形を閉じる場合の矛盾と対策

図6の ★3 の問題である．始点 a_0 は最初の点だから，その存在する空間の判定はトポロジカルな制約なしで座標値の数値計算だけで判定され，誤差の影響を受ける．次に，終点 $a_{00}(=a_0)$ の位置する空間は，直前のイベントと矛盾がないように判定される．このように判定アルゴリズムが違うため，もし図 12(i) のように a_0 の近傍に既設辺があると，両者の空間の判定が一致せず，結合不能で暴走することがある．対策としては同図(ii)のごとく，入力多角形Aを一巡後も引き続き外形沿いに a_it を伸ばし，これが a_it の既入力辺と直接接したら両者を接続し，接続点 j から a_0 までの重複部分①を削除して多角形を完成させる．



(i) 始点 a_0 と終点 a_{00} とが接続不能



(ii) 対策：Aの辺に沿って辺を伸ばし j点で接続，①の部分は消去．

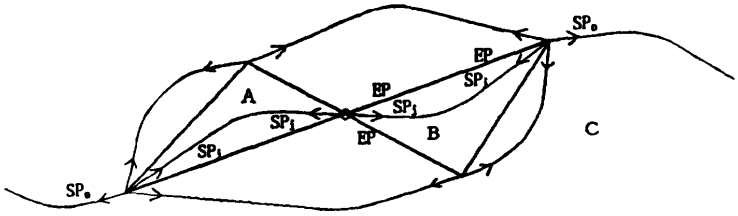
図 12 始点終点の接続：矛盾回避手法
Fig. 12 To connect start point with end point: Method to prevent inconsistency.

条件が成立するまでAの外形を何度回ってもよいから，これは既設辺が a_0 の近傍のみでなく，Aの外形全体に多重に重なるときにも有効である．

3.6 自己交叉と図形内外の判定

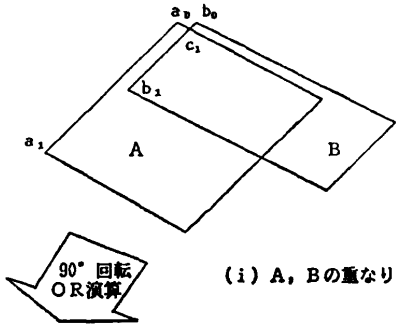
図 13 の図形は入力誤りや計算誤差で発生する．また頂点が近接した**図 14**(i)の多角形で OR を取って回転すると，計算誤差で(ii)になることがある．これらは従来方式では問題を起こすが，SPF では自己交叉の検出や図形内外判定が局所的にできるから，図13のA，B部を内部と定義したり，図 14(ii)では最外部の辺を残して内側の辺をすべて消去する等，要求に合わせた処置ができる．多角形の内外判定は，ある頂点から辺の片側に沿って一巡して数えた凸空間のポインタ SP_0 (辺間に3個の SP がはさまれている) の数を N_{SP0}，凹空間ポインタ SP_i (辺間に1個の SP) の数を N_{SPi} とし，次の関係から実行できる．

$N_{SP0} < N_{SPi}$ ……囲まれた内部空間
〃 ＞ 〃 ……囲まれない外部空間

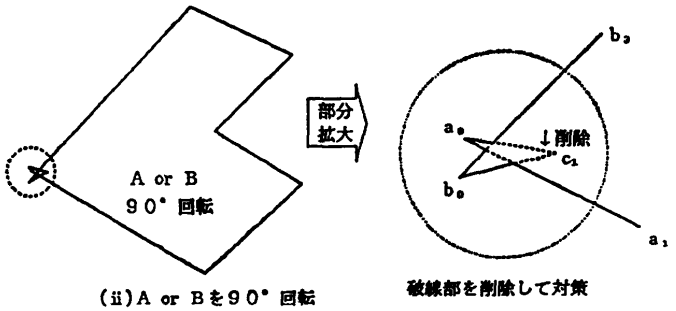


EP にはさまれた SP が1個なら SP_i，3個なら SP_0
辺に沿って数えた SP_0 数＞SP_i 数：外側空間 (C部)
〃 〃 ＜ 〃 ：内側空間 (A，B部)

図 13 自己交叉図形と図形内外の判定
Fig. 13 Self-crossing and in or outside recognition of a figure.



(i) A，Bの重なり



(ii) A or B を90° 回転　　破線部を削除して対策

図 14 座標変換誤差による自己交叉の発生
Fig. 14 Occurrence of self-crossing by coordinate transformation.

4. 実験結果

4.1 暴走対策実験プログラム

暴走対策部はC言語で約500行である．対策の影響で SPF 生成時間が一頂点当り 3.5 ms から2割方伸びて約 4 ms になったが，処理時間が平均的に一頂点当り $O(1)$，全体で $O(N)$ であることは変わらず，また SPF 生成以外は変化はない．実験では計算誤差の影響を強く出すために，座標値は短整数型 (short int) とした．

4.2 微小三角形乱数重ねによる加速暴走実験

整数 6×6 の小領域内で，小三角形の位置と回転角を乱数として SPF を生成した．座標値は整数に限られ，図形が小さいから，交点で割合として大きな切り捨て誤差がでる．**図 15** は2個の三角形の重なりの拡大表示であり，交点位置が誤差で大きく歪んでいる．図形の重なり数は8色で示している．50 個の三角形を重ねると**図 16** のような奇妙な形になる．この場合

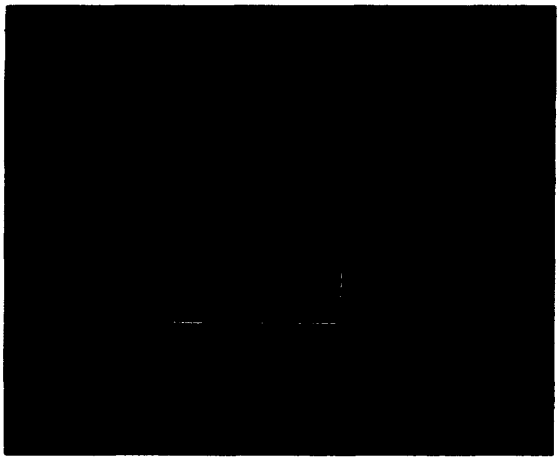


図 15 微小三角形2個の重なり表示
Fig. 15 Two small triangles.

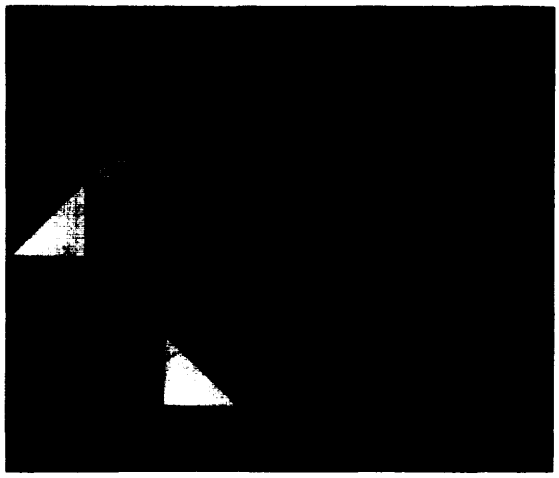


図 16 微小三角形50個の重なり表示
Fig. 16 Fifty small triangles.

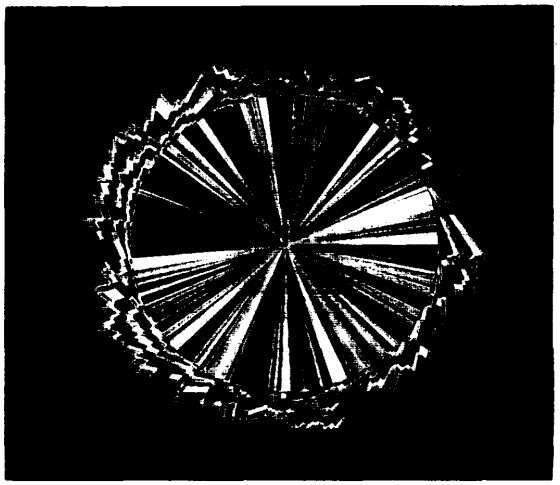


図 17 乱数で回転した三角形200個の重なり表示
Fig. 17 Randomly rotated 200 triangles.

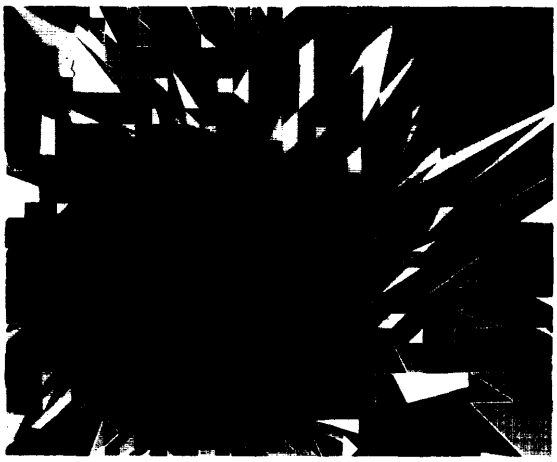


図 18 図17の中心部拡大表示（約800倍）
Fig. 18 Center of Fig. 17 (×800 oversize).

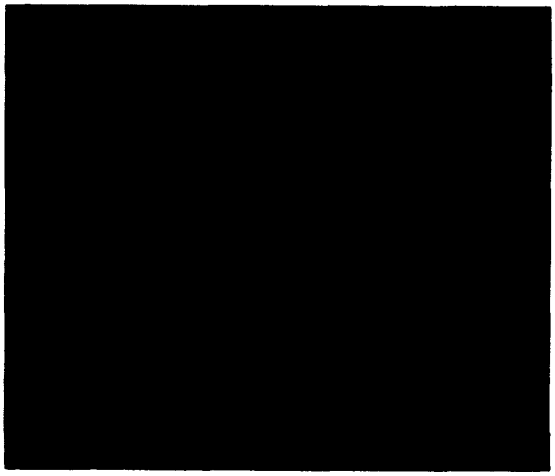


図 19 回転三角形1,000個の OR
Fig. 19 OR of 1,000 triangles.

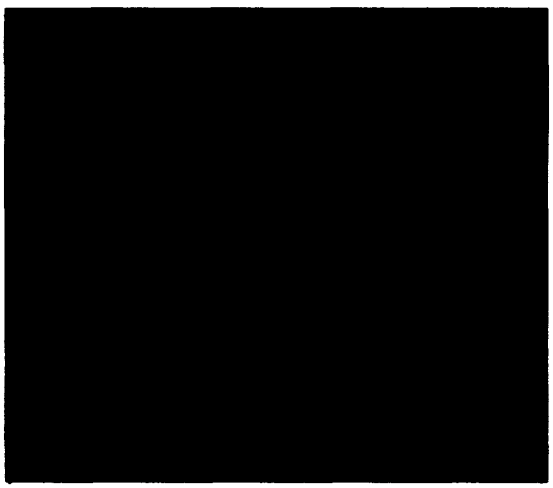


図 20 図19の中心部拡大
Fig. 19 Center of Fig. 19 (×900 oversize).

には同一座標点に数十個の頂点，交点が重なる．

次の実験で対策の効果を調べた．始端 a_0 から多角形を一巡した折線が終端に達したとき，終端の座標値は a_0 に等しいが，図 12 のごとく a_0 と同一空間に来ないことがある．その場合は3.5節の，多角形一巡後も外形に沿って辺を伸ばす対策プログラムが走る．実験ではその都度メッセージを表示した．その結果入力三角形 30 個以下では5〜10個に1個の割で，また入力50〜100個では2〜3個に1個，入力100個以上ではほとんど毎三角形ごとにメッセージが発生した．当然対策プログラムなしではメッセージ印字の代わりに暴走する．以上で従来なら確実に暴走する厳しいデータについても，対策の有効性が実証できた．

4.3 近接頂点を持つ多数三角形による加速テスト

図 17 は一頂点を半径 10 の小円上に，他の一頂点を中心が同じで半径 10,000 の大円上に置いた三角形を回転角度を乱数にして200個重ねて SPF に入力したもの．**図 18** はその中心部の 1,000 倍拡大である．拡大写真のごとく，中心付近では多数の交点が発生している．また座標値は少数点以下を切り捨てた整数値となっているため，よく見ると大変歪んだ形になっている．全交点数は 12,840 個である．実験は20種以上の類似データで行ったが，暴走は発生しない．**図 19** は同様な 1,000 個の三角形に関し OR 演算を施したもの，**図 20** はその中心部を拡大したものである．

5. まとめ

スペースモデルを使い，トポロジーに矛盾が起きぬようにデータ構造を生成する方式で，暴走防止ができることを実証した．今回は二次元多角形について検討したが，この方式は三次元を含む各種図形に適用可能と考えられる．スペースモデルでは，注目，局所図形集合演算，運動図形の衝突検出など多くの機能が平均的に $O(1)$ の対話速度で実行できるが，今回の検討で信頼性を兼ね備えることができ，実用化の基礎を固めることができた．スペースモデルの問題点は，ポインタの分だけメモリ所要量が大きい（二次元は浮動小数点座標値のみの5〜6倍，三次元はウイングド・エッジ方式の 1.5〜2 倍）ことである．しかし処理速度がオーダ違いに速いから，リソース消費（＝メモリ容量×処理時間）は従来方式より大幅に小さい．さらに処理の局所性がよいので仮想記憶のスワッピングが少なく，メリットが大きい．今後は隠線消去等簡単な用途から応用の検討を進めるつもりである．

謝辞 最後に本研究を進めるにあたり有益な御討論を頂いた東京大学計数工学科・杉原厚吉助教授，便宜を計って下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，日立京浜工業専門学院・湯沢宏学院長，他の方々に厚く御礼申し上げます．

参考文献

1) 大沢 晃：二次元スペース・モデリングの考察—ポインタ方式空間表現による局所的図形処理—，情報処理学会論文誌，Vol. 27, No. 12, pp. 1174-1185 (1986).
2) 大沢 晃，川崎敏治，岩本哲夫：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，情報処理学会論文誌，Vol. 29, No. 10, pp. 933-943 (1988).
3) 杉原厚吉，伊理正夫：計算誤差による暴走の心配のないソリッドモデラの提案，情報処理学会論文誌，Vol. 28, No. 9, pp. 962-974 (1987).
4) 伊理正夫，杉原厚吉：計算誤差を考慮した幾何的アルゴリズム，情報処理学会アルゴリズム研究会資料，1-1 (1988).
5) 伊藤，佐々木：Consistency の実現を目的とする多面体モデラの開発，情報処理学会グラフィクスと CAD 研究会資料，6-4 (1982).
6) Segal, M. and Sequin, C. H.: Consistent Calculations for Solid Modeling, *Proceedings of the ACM Symposium on Computational Geometry*, Baltimore, pp. 29-38 (June 1985).
7) Green, D. H. and Yao, F.: Finite-resolution Computational Geometry, *27th ACM Annual Symposium on Foundations of Computer Science*, Tronto, pp. 143-152 (Oct. 1986).
8) 伊理正夫：計算機何学と地理情報処理の基本，bit, 1986年9月号別冊，pp. 6-13 (1986).
9) 木村文彦：CAD/CAM の動向，日経 CG, 1987年11月号，pp. 124-130 (1987).
10) 千代倉弘明：ソリッドモデリング，工業調査会 (1985).

付録 二次元スペースモデルの原理[1,2]

多角形で説明する．**図A1** のごとく頂点の周りの凹な（辺にはさまれた角度が 180° より大な）空間を，頂点を通る Y 軸に平行な仮想線 Bx（図中一点鎖線）で区切り，Bx と各図形の辺 By によって全体を部分空間 $\omega_0, \cdots, \omega_i, \cdots$ に分割する．この場合 Bx が By を突き抜けることはないとする．そうすると各 ω_i はその X^+, X^- 端（X座標値最大および最小の位置）に必ず頂点を持つ（辺の交点も頂点に含むとする）．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

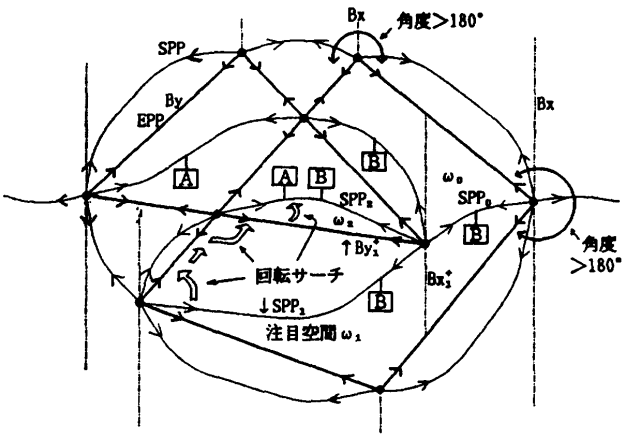
の対を SPP (Space Pointer Pair：図中矢印付細線)，SPP を設置した全体の図形ファイルを SPF (Space Pointer File) と名付ける．

X座標値の等しい複数の頂点が存在するときは，それらが相互にX方向にわずかにずれていると考えれば（考えるだけで座標値は変更しない)，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 には必ず一組の SPP_i が一対一で対応する．一対一なので，SPP_i は ω_i を表現すると言える．SPF では図形の外部の空間も内部と同様に SPP で表現される．内部と外部を区別する情報は，図形の色彩や図形の名前等と共に SPF 生成時に図A1の A B のごとく SPP に属性として付加される．これを空間属性と呼ぶ．これにより，図形の重なりは SPPの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もつ空間を出力すればよい．SPF 上で一つの多角形が与えられたとき，これと他の多角形との重なり部分の形状を出力する処理を「局所的 AND」と名付ける．その場合の平均的な処理時間は全体の図形数と無関係で $O(1)$ となる．

SPF では図形辺 By もポインタ対 EPP (Edge Pointer Pair) で表現し，各頂点（交点を含む）には座標値 (X, Y) および，SPP, EPP の片側を構成するポインタ EP, SP を内容とする頂点テーブルを設置する（**図 A2**)．頂点テーブルの集合体が SPF である．

SPP が空間を，EPP が辺を表現しているから，図形（空間）検索問題は，SPF の中から SPP や EPP を検索する問題に置換できる．例えば図A1 で SPP_1 を与えたとき，ω_1 の X^+ 側の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 SPP_1 の空間の Y^+ 側外形線 By_1^+ は，SPP_1 から図中の白抜き矢印 ↶ のようにポインタをたどる「回転サーチ」で求められる．さらに，Y^+ 側隣接空間を表現する SPP（図中 A，B の重なり部）も，By_1^+ からの回転サーチで検索できる．回転サーチは局所的で，特に意地悪な形状を除けば，全体図形数に無関係な $O(1)$ の処理である．

このようにX側，Y側共に隣接空間が高速に検索できるから，プログラムは人間のように，隣接空間伝いに自由な方向に注目点を移動することができる．**図 A3** には任意の空間 SPP^0 から出発して，座標値で与えられた目標点 a_0 を含む空間（$SPPa_0$ で示す）へ注目点を移動する方法を示す．

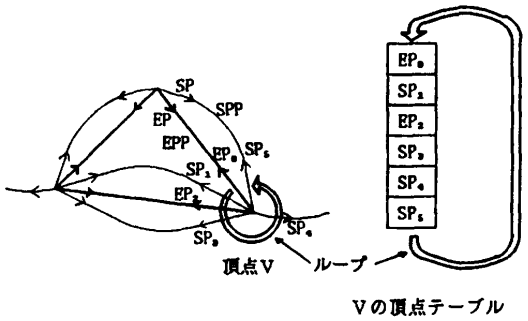


A B は色，名前などを表す空間属性

Y方向の検索は白抜き矢印の回転サーチによる

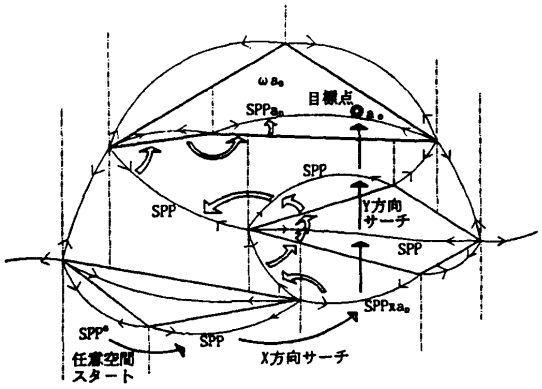
図A1 スペースモデルのデータ構造 SPF

Fig. A1 Data structure SPF of space model.



図A2 頂点を表現するループ状頂点テーブル

Fig. A2 Loop wise vertex table represents a vertex.

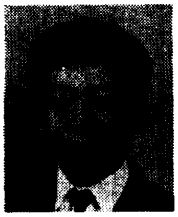


Y方向の検索は白抜き矢印の回転サーチによる

図A3 目標点 a_0 を含む空間 $SPPa_0$ の探索

Fig. A3 Retrieval of subspace $SPPa_0$ including target point a_0.

(昭和63年11月22日受付)
(平成元年10月11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．