Vol. 32 No. 6 情報処理学会論文誌 June 1991

数値的に安定な分割統治型 Voronoi 図構成算法†

大石泰章†† 杉原厚吉††

Katajainen and Koppinen の研究によって理論的にも実用的にも最も速い Voronoi 図構成算法となる可能性を持つに至った分割統治型算法を，計算の途中で数値誤差が混入しても破綻しないように改良した．理論的に正しさが保証された算法であっても，数値誤差の生じる現実の計算機では破綻することがある．この破綻を防ぐために，数値計算結果が対象の位相的性質と矛盾しそうになったら処理の方向を転換して矛盾を防ぐという形の歯止めを算法の中に挿入する，というのが改良の基本的な方針である．この歯止めは，①数値誤差が発生しても処理が行き詰まらないことを保証し，②数値計算が正しいときは算法の動きを妨げず，③計算量を著しく悪化させないものでなくてはならない．本論文では，この3条件を満たす歯止めをかけることによって，分割統治型算法を数値的に安定なものにした．この新しい算法に基づいて作ったプログラムはどんなに低精度で計算しても破綻せず，いままで破綻してしまって求められなかったような種類の Voronoi 図も正しく求めることができた．また歯止めをかけても計算量が悪化しないことも確かめられた．

1. はじめに

Voronoi 図は Euclid 平面上に与えられた有限個の点（以下これを母点という）に対してその勢力圏を表す図で，多くの応用分野を持つ重要な幾何的概念である[4),10),11)]．

Voronoi 図の構成算法は逐次添加法[2),10)]と分割統治法[11)]の二算法が有力である[4)]．これまでは点の数が N 個のとき平均的な計算量が $O(N)$ になる逐次添加法が最も実用的な算法であるとされてきた[7)]．しかし最近 Katajainen and Koppinen によって分割統治法が改良され，従来 $\Theta(N \log N)$ だった平均的計算量[7)]が逐次添加法と同程度の $O(N)$ に引き下げられた[5)]．もともと最悪の場合の計算量では逐次添加法の $\Theta(N^2)$ に対して分割統治法は $O(N \log N)$ で，分割統治法の方が優れていたため[6)]，逐次添加法をしのぐ可能性もあるとして今や分割統治法はその魅力を増している．

ただし以上の話は「計算の途中では数値誤差が混入しない」という仮定をおいた理想の世界での話である．これに対して実際に計算機上で算法を実行するときには数値誤差の混入がまぬがれえないため，理論どおりに Voronoi 図が求められるわけではない．実際，経験的に分割統治法では単精度計算で 1,000 点，倍精度計算でも 3,000 点が暴走しないで結果を求められる限界だと言われてきたようである[12)]．

杉原・伊理はこの問題を根本的に解決するために逐次添加型の Voronoi 図構成算法を改良し，「どんなに計算精度が低くても破綻せず，正しい Voronoi 図が満たすべき位相的性質を少なくとも満足する結果を出力し，計算精度を上げていくにつれて出力が正しい Voronoi 図に『収束』していく」という新しい算法を構成した[12)]．この新しい算法は数値誤差が混入することを前提としているため，退化が生じていても単に退化に近い状態として扱うことができ，例外処理が不要になるという副産物も得られる．ところで杉原・伊理が逐次添加法を取り上げたのは，当時この算法が最も実用的とされていたからである．一方，上で述べたように分割統治法が理論面で改良されてきたために，分割統治法に対して同様の数値的安定化が可能か否かを確かめることは大きな意義を持つに至っている．

本論文では以上の動機に基づいて分割統治型 Voronoi 図構成算法を上の意味で数値的に安定にする[8),9),13)]．まず第2章で安定化の基本方針を示し，各種の準備を行う．それをふまえて第3章で分割統治型算法を紹介し，これを数値的に安定にする．第4章では前章で行った安定化の正当性を証明し，第5章では安定化された算法の持つ良い性質を実験によって実証する．

2. 数値的安定化の基本方針

2.1 数値的安定化

Voronoi 図構成に限らずある幾何的アルゴリズムを前章の意味で数値的に安定にする方法は，次のように整理できる．すなわちこうした算法の多くは扱う図形の位相構造をグラフなどの組合せ論的方法で表現し，図形の計量的性質に関する浮動小数点計算の結果

† Numerically Robust Divide-and-Conquer Algorithm for Constructing Voronoi Diagrams by YASUAKI OISHI and KOKICHI SUGIHARA (Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo).

†† 東京大学工学部計数工学科

にしたがって，この位相構造を変更していくことで構成されている．ここで数値誤差の混入で算法が破綻するのは，誤差のために浮動小数点計算が誤差分だけずれた結果を出し，そのために誤った分岐をして誤った位相構造変更を行ってしまうからである．だから破綻を防ぐためには，数値計算が正しければ必ず満たされるべき位相的性質をいくつか取り上げ，これと矛盾する位相構造変更がなされないか否かを監視し，なされようとしたときは浮動小数点演算の結果より位相的性質の保存を優先することによって最低限算法が破綻しないことを保証すればよい．矛盾に陥らないためのこの監視手続きを以下では位相的歯止めと呼ぶ．位相的性質の監視や位相構造の変更は組合せ論的演算のみによって行われるのでそれ自体が数値誤差によって誤る心配はない．したがって位相的歯止めをかけることで矛盾の発生は完全に防止できる．

まとめると，かけるべき位相的歯止めは算法に次の性質を与えるものである．

(Q1) 計算精度に関わらず（どれほど大きな数値誤差が発生しても）算法は有限回の動作の後に結果を出力して終了する．

(Q2) 計算が正しいときには，（位相的歯止めは算法の動きに実質的な影響を与えず）歯止めをかける前と同じ結果を出力する．

(Q3) 計算が正しいときには，歯止めをかける前と比べて計算量が著しく悪化することがない．

ただし「計算が正しい」とは，すべての浮動小数点計算を無限桁の精度で行ったときと同じ分岐をたどって算法が実行されることであるとする．

なお，以上の条件を満たす限り位相的歯止めはできるだけ厳しいことが望ましい．

2.2 Voronoi 図（Delaunay 網）の位相的性質

Voronoi 図の厳密な定義等詳しいことは文献 4), 11) 等を参照されたい．Voronoi 図の例を図1(a)に示す．

二つの母点の勢力圏が隣り合っているとき，この二つの母点を線分でつなぐことにする．このとき得られる図を Delaunay 網という．例を図1(b)に示す．Delaunay 網を構成することは Voronoi 図を構成することと同等であり，前者の立場でみた方が数値的安定化がしやすいと思われる[8]ので本論文ではDelaunay網構成の側から算法を論じることにする．

数値的安定化のために用いる Delaunay 網の位相的性質として次のものに注目する．

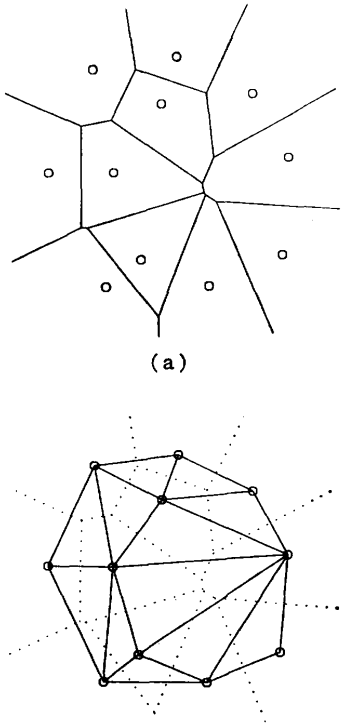


図 1 Voronoi 図と Delaunay 網（小丸は母点の位置を表す）
(a) Voronoi 図，(b) Delaunay 網
Fig. 1 Voronoi diagram and the corresponding Delaunay net (the generators are represented by small circles).
(a) Voronoi diagram, (b) Delaunay net

(D1) Delaunay 網はグラフとして連結である．

(D2) Delaunay 網はグラフとして並列枝を持たない．

(D3) Delaunay 網は平面の分割である．

(D4) Delaunay 網の有限な面はすべて三角形である．

また計量的な性質として Delaunay 網が与えられた母点の凸包の分割となることにも注目する．

2.3 Voronoi 図（Delaunay 網）の位相構造の表現

本論文では対象図形の2次元複体としての位相構造を Guibas and Stolfi によって提唱された方法[3]で表現する．この方法による表現を以下では図形の GS 表現と呼ぶ．この方法では枝に向きをつけたもの（二つの付け方がある）を基本的なオブジェクトとし（以下これを有向枝と呼ぶ），各頂点の周りの有向枝と各面の周りの有向枝を循環リストの形にして記憶する．

定義 2.1 有向枝 a に対して次の関数を定める.

(1) sym(a): a と同じ無向枝に属し, a と逆の向きを持つ有向枝.

(2) onext(a): a の始点の周りに反時計回りに回ったとき最初に出会う無向枝に, a の始点を始点とする向きをつけた有向枝.

(3) oprev(a): a の始点の周りに時計回りに回ったとき最初に出会う無向枝に, a の始点を始点とする向きをつけた有向枝.

(4) org(a): a の始点.

(5) dest(a): a の終点.

(6) left(a): a の向きに向かって左の面.

(7) right(a): a の向きに向かって右の面. □

さらに GS 表現の構造変更に用いる手続きを定義する.

定義 2.2 left(a)=left(b) を満たす有向枝 a, b に対して, 次の手続きを定める.

(1) connect(a, b): 新しい有向枝 c を, org(c)=dest(a), dest(c)=org(b), left(c)=left(a)=left(b) を満たすように現在の GS 表現に付け加え, 値 c を返す.

(2) deleteedge(a): 有向枝 a の属する無向枝を現在の GS 表現から開放除去する. 値は返さない. □

任意に構成した GS 表現は必ずしも平面の分割として実現可能とは限らない. しかし,

補題 2.1 平面の分割として実現可能な GS 表現(必ずしも連結でなくてよい)とそれに属する有向枝 a, b において left(a)=left(b) であれば, connect(a, b) の実行後も平面の分割として実現可能である. □

ただしある GS 表現が連結であるとは, その GS 表現に属する任意の 2 本の有向枝が各頂点の周りの循環リストまたは各面の周りの循環リストまたはその両方を介して一方からもう一方へ到達可能であることをいう.

次にある GS 表現が平面の分割として実現可能であるとき, ある面を無限領域にすると指定しておけば実現の方法は一意に定まる (文献 3) の Theorem 3.7 より). 面を指定するにはその面が left(a) になるような有向枝 a を指定すればよい. この有向枝を外周枝, 無限領域の面を取り巻く閉路を外周と呼ぶ.

さらに外周枝が指定されている GS 表現において, 外周上に R, L, T, B なる四つの母点 (以下まとめて四端点と呼ぶ) を指定したものを GS 網と呼ぶ. $R(L)$ は属する GS 表現の母点の中で x 座標が最大〔最小〕の母点 (複数ある場合にはその中でさらに y 座標も最大〔最小〕の母点. 以下同様) とし, $T(B)$ は y 座標が最大〔最小〕の母点であるとして定めるが, 計算精度が低いときには以上の性質が必ずしも本当に満たされるとは限らない.

2.4 計量的な関数

Voronoi 図 (Delaunay 網) 構成に用いる浮動小数点計算のほとんどは本節で述べるただ二つの論理関数に集約できる.

定義 2.3 母点 A, B, C, D の正規直交座標を (x_A, y_A) のように書くとき,

(1) ccw(A, B, C): 行列式

$$\begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix}$$

の値が正のとき True, それ以外のとき False を返す.

(2) incircle(A, B, C, D): 行列式

$$\begin{vmatrix} x_A & y_A & x_A^2+y_A^2 & 1 \\ x_B & y_B & x_B^2+y_B^2 & 1 \\ x_C & y_C & x_C^2+y_C^2 & 1 \\ x_D & y_D & x_D^2+y_D^2 & 1 \end{vmatrix}$$

の値が正のとき True, それ以外のとき False を返す.

とする. □

十分計算精度が高ければ, ccw(A, B, C)=True は母点 A, B, C がこの順に反時計回りに位置していることと同値であり, incircle(A, B, C, D)=True は ccw(A, B, C)=True のとき母点 A, B, C を通る円の内部に母点 D が位置していることと同値である.

3. 分割統治型算法とその数値的安定化

3.1 算 法

分割統治型算法では, まず母点集合を十分小さな部分集合に分割してそれぞれに対する Delaunay 網を作り, 次に二つの Delaunay 網を融合させて, 一つの Delaunay 網にする手続き (これを以下ではマージという) を繰り返す. 算法の本質的部分であるマージの手続き HorizontalMerge を図 2 に示す. 図 2 の手続きはマージする二つの Delaunay 網の母点数が共に 2 点以上であるときのみを扱っているが, それ以外のときの手続きもこれに準じて定めるものとする. またマージ以外の部分については文献 5) の算法に準じる

ものとする（以下これを「考えている分割統治型算法」と呼ぶ）．

以下図2の手続きについて解説する．ここでは{…}以外の部分が従来の手続きを表し，{…}で示した部分が数値的安定化のために新たに追加した部分を表す．/*…*/で示した部分はコメントである．この手続きはマージすべき二つの Delaunay 網を GS 網 F, S として入力すると，これらのマージ結果として GS 網 M を出力する．ただし計算が正しいときはFに属す母点の x 座標は S に属す母点の x 座標より小さいと仮定する．

従来の手続きでは，最初にFとSの下側および上側共通接線をみつけ，マージのために新しい枝を付加すべき範囲を設定する (H6〜H9 行目)．手続きの主要部分では，まず下側共通接線上の枝を付加し，それを有向枝 basel と呼ぶ (H10 行目)．次に basel の上側の三角形を構成する枝の初期の候補 cand1 と cand2 をそれぞれ F と S から選び (H11, H12 行目)，F 内，S 内の不用な枝を除きながらこれらの候補を更新する (H13〜H21 および H22〜H30 行目)．二つの候補枝の一方 winner を選んで新しい三角形を構成し，そのとき付加した枝を新たに basel とする (H32, H36 行目)．これを上側共通接線に到達するまで繰り返す．H31 行目は上側共通接線に到達したか否かの判定で，H38〜H41 行目は結果の出力のためのものである．

両端点がともにFの母点であるような枝を F-F の枝と呼ぶ．S-S の枝，F-S の枝も同様とする．さらに以下では周囲の頂点がすべて F(S) の母点であるような面のことを F(S) の面，F の面でもSの面でもない有限領域の面を FS の面と呼ぶ．

図3に H11〜H36 行目のループを1回回る間の実例を示した．ただしこ

```
    procedure HorizontalMerge
    入力：母点数が各々2点以上であるようなGS網F,S. 但しFとSは計算が正しいときy軸に
        平行なある直線を境に，各々それより左，それより右に位置している．
    出力：FとSのマージ結果を表すGS網M.
    begin
        /*** マージ後の四端点の決定 ***/
        /* Fの四端点を R1,L1,T1,B1，Sの四端点を R2,L2,T2,B2 とする */
H1      R := R2;   L := L1;
H2      if B1 is higher than B2 then B := B2 else B := B1;
H3      if T1 is higher than T2 then T := T1 else T := T2;

        /*** shell の構成 ***/
H4      {Fの外周上で時計回りの向きを持った有向枝に "shell-edge" のマークをつける；}
H5      {Sの外周上で反時計回りの向きを持った有向枝に "shell-edge" のマークをつける；}

        /*** 共通接線の探索 ***/
H6      下側共通接線が通過する母点でFに属するもの ST1 とSに属するもの ST2 の対をみつける；
H7      上側共通接線が通過する母点でFに属するもの TM1 とSに属するもの TM2 の対をみつける；
H8      ST1 〔ST2〕を始点とする外周上の有向枝で時計〔反時計〕回りのものを st1〔st2〕とする；
H9      TM1〔TM2〕を始点とする外周上の有向枝で反時計〔時計〕回りのものを tm1〔tm2〕とする；

        /*** マージの主要部 ***/
H10     basel:=connect(sym(st2), st1);
        do /* This is merge loop */
H11         cand1:=onext(sym(basel));
H12         cand2:=oprev(basel);

            /*** F-F の枝の削除 ***/
H13         while cand1 ≠ tm1
H14             and ccw(dest(basel),org(basel),dest(cand1))
H15             and incircle(dest(basel),org(basel),dest(cand1),dest(onext(cand1)))
H16             {and cand1 isn't marked "shell-edge"} do
            begin
H17             {mark sym(onext(cand1)) "shell-edge";}
H18             {mark oprev(sym(cand1)) "shell-edge";}
H19             tmp:=onext(cand1);
H20             deleteedge(cand1);
H21             cand1:=tmp;
            end;

            /*** S-S の枝の削除 ***/
H22         while cand2 ≠ tm2
H23             and ccw(dest(basel),org(basel),dest(cand2))
H24             and incircle(dest(basel),org(basel),dest(cand2),dest(oprev(cand2)))
H25             {and cand2 isn't marked "shell-edge"} do
            begin
H26             {mark sym(oprev(cand2)) "shell-edge";}
H27             {mark onext(sym(cand2)) "shell-edge";}
H28             tmp:=oprev(cand2);
H29             deleteedge(cand2);
H30             cand2:=tmp;
            end;

            /*** マージ終了の判定 ***/
H31         if cand1=tm1 and cand2=tm2 then goto Finish;

            /*** 付加すべき枝の決定 ***/
H32         if incircle(dest(cand1),dest(basel),org(basel),dest(cand2)) then winner:=cand2
            else winner:=cand1;

            /*** tm1,tm2 の示す範囲をはみ出さないように調整 ***/
H33         {if cand1=tm1 then winner:=cand2;}
H34         {if cand2=tm2 then winner:=cand1;}

            /*** 並列枝ができないように調整 ***/
H35         {if Parallel(F,S,winner) then
                if (cand1=tm1 and winner=cand2) or (cand2=tm2 and winner=cand1) then goto Finish
                else alter the winner;}

            /*** 新しいF-Sの枝を付加 ***/
H36         if winner=cand1 then basel:=connect(sym(basel),sym(cand1))
            else basel:=connect(cand2,sym(basel));
        od; /* end of merge loop */

        Finish:;
        /*** マークの消去 ***/
H37     {すべての "shell-edge" のマークを消去する；}

        /*** 終末処理 ***/
H38     得られたGS表現をMと名付ける；
H39     tm2 をMの外周枝に指定する；
H40     R,L,T,B をMの四端点に指定する；
H41     return M;
    end;
```

図 2 手続き HorizontalMerge
Fig. 2 Procedure HorizontalMerge.

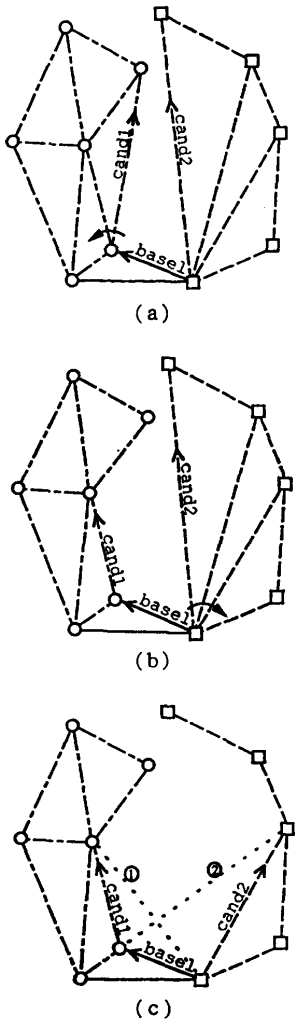


図 3 手続き HorizontalMerge の振舞いの例
(a) H13～H21 行目, (b) H22～H30 行目, (c) H32～36行目, 新しい basel は winner=cand1 のときは①のように, winner=cand2 のときは②のように付加される

Fig. 3 Example of the behavior of the procedure HorizontalMerge.
(a) ll. H13-H21, (b) ll. H22-H30, (c) ll. H32-H36, if winner=cand1 a new basel is added as ① or else as ②.

ここではFの母点を小丸で，Sの母点を小四角形で，F-F の枝を一点鎖線，S-S の枝を破線，F-S の枝を実線で表している．

3.2 数値的安定化

考えている分割統治型算法を数値的に安定にするために，(C1)～(C5) の五つの位相的歯止めをかける．

まず十分小さな母点集合（2個以下とする）に対する初期 Delaunay 網を作る段階での歯止めである．

(C1) GS 網Gを母点集合から直接生成するときには次の五つの条件を満たすように作る．

(C1.1) Gは連結である．

(C1.2) Gに並列枝は存在しない．

(C1.3) Gは平面の分割として実現可能である．

(C1.4) Gの有限領域となる面はすべて三角形である．

(C1.5) Gの四端点は外周上で反時計回りに B, R, T, L の順に位置し，Gが singleton である場合を除いて $R \neq L$, $T \neq B$ である． □

これはGが singleton のときは枝は引かずに唯一の母点を $R=L=T=B$ として定め，Gが母点2個からなるときはそれらを結ぶ枝を1本引いて一方の母点を $R=T$, もう一方を $L=B$ と定めるか，あるいは一方を $R=B$, もう一方を $L=T$ と定めるかどちらかだということになる．

2番目は共通接線発見法に関する歯止めである．

手続き HorizontalMerge の H6 行目で下側共通接線は次のようにして求める．まず，FとSの外周上の母点から適当に1点ずつ選んで，それぞれ ST_1, ST_2 とする．この2点を結ぶ直線が最初の共通接線候補となる．さてFの外周上で ST_1 の両隣りにある2点を調べ，現在の共通接線候補より下にあるものがないか見る．もしそうした母点があれば，それを新しい ST_1 にして共通接線候補を更新する．同様に S の外周上でも ST_2 の両隣りの母点について調べ，必要なら共通接線候補を更新する．以下この操作を繰り返して収束したものを下側共通接線とする．上側共通接線についても同様である．ある2点を結ぶ直線に関して第3点がどちらの側にあるかは，関数 ccw を用いて判定できる．しかしこの関数は数値誤差の影響を受けるから歯止めが必要である．

下側共通接線の発見法を例にとる．

(C2) 下側共通接線の接点の探索は**図 4** (a)(b) に示す範囲で行い，(c)に示すように ST_1〔ST_2〕を時計〔反時計〕回りに動かした後は，(ST_2〔ST_1〕を動かして状況が変化しても) ST_1〔ST_2〕を反時計〔時計〕回りに後戻りさせて動かさない． □

上側共通接線についても同様の歯止めをかける．

3番目は枝の削除に関する歯止めである．

まず前処理として H4〔H5〕行目で，F〔S〕の外周上の有向枝で時計〔反時計〕回りの向きのものに "shell-edge" とマークする．**図 5** に例を示す．太い線

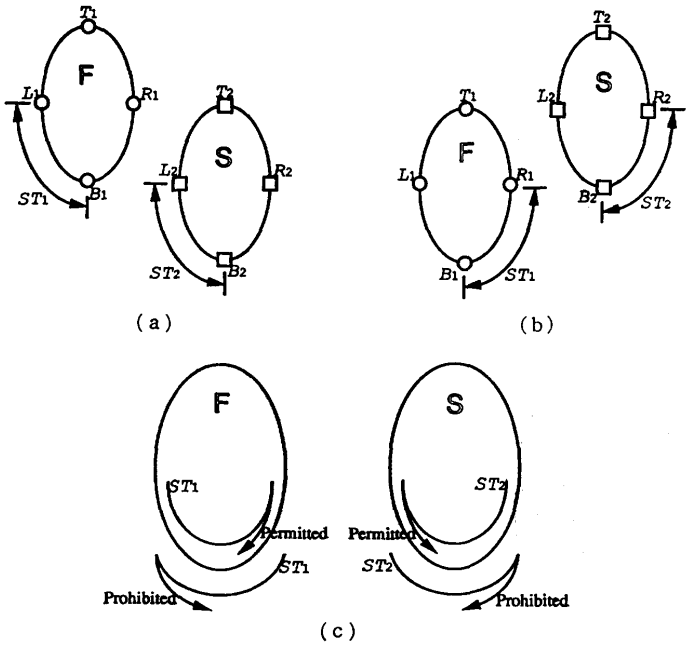


図 4 下側共通接線の接点候補 ST_1, ST_2 の更新法の制限
(a) B_1 の方が B_2 より y 座標が大きいとき, (b) B_2 の方が B_1 より y 座標が大きいとき, (c) 動かし方の制限

Fig. 4 Restrictions on the way to change candidates ST_1 and ST_2 for the points of tangency of the lower common tangent.
(a) case where B_1 has a larger y-coordinate than B_2,
(b) case where B_2 has a larger y-coordinate than B_1,
(c) restriction on movements of ST_1 and ST_2.

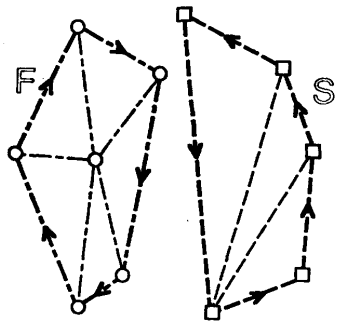


図 5 shell の構成

Fig. 5 Construction of the shells (directed edges represented by arrows belong to the shells).

で描かれているのがマークされた枝である. 以下このマークを持つ有向枝によって誘導される有向グラフを shell と呼ぶ. 枝を削除するごとに H17, H18, H26, H27 行目で shell の更新を行い, 次の歯止めをかける (H16, H25 行目).

(C3) 有向枝 cand1, cand2 が "shell-edge" とマークされているときはこれを削らない. □

4 番目は枝の付加に関する歯止めである.

(C4) 並列枝を生じるような枝は付加しない. □ H35 行目の論理関数 Parallel は winner の値が今のままでは H36 行目で並列枝を生じてしまうというとき True を返す. このときは winner の値を変えて付加を行う. 扱っている GS 表現が平面グラフとして実現可能ならば, winner の値を cand1 にしても cand2 にしても並列枝を生じてしまうということはないので, これによって並列枝の発生を防止できる.

最後は枝を付加する範囲に関する歯止めである.

(C5) 有向枝 tm1, tm2 によって指定される「枝を付加する範囲」をはみ出す付加は行わない (H33, H34 行目). □

winner の値を cand1 にしても cand2 にしても (C4) か (C5) のどちらかに反するときには, goto Finish を実行してマージ自体を終わらせる (H35 行目).

4. 数値的安定性の証明

4.1 条件(Q1)の充足

本章では前章でかけた五つの歯止めにより算法が数値的に安定になること，すなわち条件（Q1）～(Q3) を満足することを順に示していく．

本節では条件（Q1）を満足することを示す．

補題 4.1 性質（C1.1)～(C1.5) を満たす GS 網 FとSを，手続き HorizontalMerge に与えるとき，計算精度にかかわらず H12 行目を実行した段階では常に次の性質が成り立つ．

（1） マージされつつあるFとSはグラフとして一つの連結成分からなる．

（2） shell は有向グラフとして二つの連結成分からなり，連結成分はおのおの有向閉路をなす．

（3） shell の二つの連結成分のうち，一つは F-F の枝のみ，もう一つは S-S の枝のみからなる（おのおのFの shell，S の shell と呼ぶ)．

（4） F-F の枝〔S-S の枝〕aにおいて，aがFの shell〔S の shell〕に入っていることと left(a)〔right (a)〕がFの面〔Sの面〕でないこととは同値．

（5） sym(cand1) はFの shell に，sym(cand2) はSの shell に属する．

（6） マージされつつあるFとSには並列枝は存在しない．

（7） org(basel) はSの母点，dest(basel) はFの母点，left(basel) は FS の面（ただし H12 行目を初めて実行するときに限って無限領域)，right(basel) は無限領域である．

（8） マージされつつあるFとSの有限な面はすべて三角形．

（9） マージされつつあるFとSのGS表現は平面の分割として実現可能である．

証明 初めて H12 行目を実行するとき(1)～(9) が成立することは算法により明らか．

まず cand1 を削除するのはこれが shell に属さないときだから，(4)により left(cand1) はFの面である．よって削除後も(1)が保存され，その後の更新手続きによって(2)～(5)も保存される．cand2 の削除においても同様である．

枝を付加する際も(9)が成立しているから H35 行目の調整によって(6)が保存される．(7)～(9)が保存されることも算法からいえる．

以上により題意がいえた． □

補題 4.2 性質（C1.1)～(C1.5) を満たす GS 網 FとSを手続き HorizontalMerge に与えるとき，計算精度にかかわらず HorizontalMerge は有限回の操作の後に終了して，出力Mは（C1.1)～(C1.5) を満たす．

証明 まず H6～H9 行目の部分を考える．ここが有限時間で終わるのは，(C2) により接点探索が図4(a)(b)の領域を高々1往復して終わることにより明らか．

次に H10～H36 行目のループの部分では，補題 4.1 により破綻は起こらない．よって H31 行目か H35 行目の goto Finish が有限時間の後に実行されることをいえばよいが，補題 4.1(2)(5)の成立と sym(tm1), sym(tm2) が shell に属することに注意すれば算法よりこれがいえる．

ループを抜けた後も補題 4.1 (1)(6)(9)(8) が成立するので出力 M が (C1.1) (C1.2) (C1.3) (C1.4) を満たすことがおのおの導かれ，枝を付加する範囲を示した st1, tm1 等の有向枝が歯止め (C2) のもとで選ばれていることから (C1.5) も保証される． □

歯止め（C1）をかけると，この補題 4.2 をマージのたびに適用できるので次がいえる．

定理 4.1 考えている分割統治型算法は有限回の操作の後に，(C1.1)～(C1.5) を満たす GS 網を出力して終了する． □

4.2 条件(Q2)の充足

定理 4.2 計算が正しいとき，歯止め (C1)～(C5) をかけても，考えている分割統治型算法は与えられた母点に対する Delaunay 網を出力する．

証明 歯止めをかける前の算法が与えられた母点に対する Delaunay 網を正しく構成することは，文献 3) の Lemma 9.5 が保証している．よって計算が正しいときにはこれらの歯止めが実質的に算法の動きに影響しないことをいえばよい．

まず歯止め (C1) は正しい Delaunay 網が (D1)～(D4) を満たし，与えられた母点の凸包の分割であることにより影響を与えない．

次に歯止め (C2) を考える．正しい Delaunay 網は凸包の分割であるから接点が存在するのは明らかに図4(a)または(b)の範囲である．また (C2) の後半の条件も実質的には算法に影響しない．なぜなら，例えば図6において ST_1 を時計回りに動かした (①→②) 後，ST_2 の移動があって (②→③) 状況が変化し

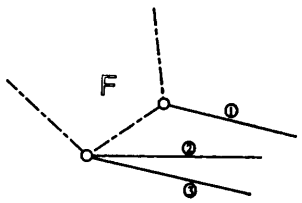


図 6 図 4(c) に示した動かし方の制限が正当である理由
Fig. 6 Correctness of the restriction on the movements of ST_1 and ST_2 shown in Fig. 4(c).

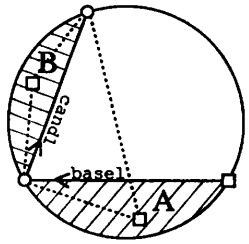


図 7 有向枝 cand1 が削られるときの母点 dest(onext(cand1)) の存在範囲
Fig. 7 Region in which the generator dest(onext(cand1)) lies in the case where cand1 is to be deleted.

たとしても，このとき ST_2 は ST_1 のまわりを時計回りに移動するから，反時計回りに見て ST_1 の隣りに位置する母点（移動する前の ST_1）が現在の共通接線候補より上にあるという性質は保たれる．したがって ST_1 を再び反時計回りに戻す必要が生じることはない．他の場合も同様である．

さらに歯止め (C3) を考える．cand1 の削除について考えると，cand1 が shell に属しているのに H13 行目～H15 行目のそのほかの条件がすべて満たされることは起こらない．なぜならそうしたことが起こったとすると，この場合も補題 4.1 の(4)(8)が成立するから left(cand1) は無限領域か三角形の FS の面 (cand1 上にない第 3 の母点が S の母点) かどちらかである．無限領域であるとき，cand1 を削るとFの上半分がマージ中のFとSから非連結になってしまい，算法にしたがうと以後この非連結性は決して回復されない．これは正しい Delaunay 網が連結であること (D1) に反する．left(cand1) が上に述べたような三角形の面であるときは，incircle が True だから母点 dest(onext(cand1)) すなわち「この三角形の頂点で cand1 上にないSの母点」は，**図 7** の円の内部にあり，計算が正しいから計量的にも left(cand1) は cand1 のすぐ左にあるため，さらにこの母点は領域AかBのどちらかにある．Aにあるとすると，考えている三角形が basel と交わってしまうが，正しい Delaunay 網は母点以外の点で枝が交わることはないのでこれは矛盾．次にBにあるとすると，Fの母点とSの母点とがいかなる直線をもってしても分離できなくなってしまい，これはFとSの与え方に反する．同様のことが cand2 についてもいえるので (C3) は算法に影響しない．

歯止め (C4) については，正しい Delaunay 網は並列枝を持たない (D2) のでこの歯止めも実質的に影響しない．

最後に歯止め(C5)だが，計算が正しいとき org(tm1), org(tm2) は上側共通接線の接点になることと，Delaunay 網が与えられた母点の凸包の分割であることから，「はみ出す付加」は行われない．

以上で題意がいえた． □

4.3 条件(Q3)の充足

補題 4.3 性質 (C1.1)～(C1.5) を満たす GS 網 FとSを手続き HorizontalMerge に与えるとき，計算精度にかかわらず，Fの母点数とSの母点数との和を n とするときの共通接線を求める手間は，どちらも $O(n)$ で抑えられる．

証明 下側共通接線について考える．(C1.2) (C1.3) (C1.4) によりFもSも平面中の三角形網であるから，Fの枝もSの枝もそれぞれその総数は $3n$ 以下である（例えば文献 4）参照）．よってFの shell もSの shell も長さが $3n$ 以下の有向閉路である．接点の探索は shell 上の図 4(a)(b)に示した領域を高々1往復するだけだから，最悪でも $O(n)$ の手間で終了する．上側共通接線についても同様である． □

補題 4.4 性質 (C1.1)～(C1.5) を満たす GS 網 FとSを手続き HorizontalMerge に与えるとき，計算精度にかかわらず，Fの母点数とSの母点数との和を n とするときの論理関数 Parallel の評価に要する手間の総計は，$O(n \log n)$ で抑えられる．

証明 マージのどの段階でも補題 4.1 の(8)(9)が成立するから，マージされつつあるFとSの中に含まれる枝の総数は常に $3n$ 以下である．特に F-S の枝も常に $3n$ 本以下である．よって1回の Parallel の評価は $3n$ 本以下の枝を調べれば終了し，各母点に対して，その母点を端点とする F-S の枝を平衡木の形で記憶しておけば，調べる手間は $O(\log n)$ で抑えられる．また F-S の枝が $3n$ 本以下ということは Par-

allel の評価回数も $3n$ 回以下ということであり，これより Parallel の評価に要する手間の合計は $O(n \log n)$ で抑えられる． □

補題 4.5 性質 (C1.1)～(C1.5) を満たす GS 網 F と S を手続き HorizontalMerge に与えるとき，計算精度に関わらず，F の母点数と S の母点数との和を n とするときの手続き HorizontalMerge に要する手間は，$O(n \log n)$ で抑えられる．

証明 F の母点数と S の母点数との和が n のとき前補題と同様の理由によって，もともとある F-F の枝と S-S の枝はどちらも $3n$ 本以下であり，最終的に付加される F-S の枝も $3n$ 本以下である．補題 4.2 の(3)(5)(7)により，削られるのは F-F の枝または S-S の枝のみであり，付加されるのは F-S の枝のみである．よって削る枝の数も付加する枝の数もそれぞれ $O(n)$ で抑えられる．1本枝を削るのにかかる手間は定数であり，1本枝を付加するのにかかる手間も Parallel の評価の手間を除けば定数である．以上より HorizontalMerge の H11 行目～H36 行目のループで必要とされる手間は Parallel の評価の手間を除けば合計 $O(n)$．補題 4.3 と補題 4.4 の結果を合わせると，結局 $O(n \log n)$ で抑えられる． □

定理 4.3 計算精度にかかわらず，与える母点数を N とするときの，考えている分割統治型算法の数値的安定化後の計算量は最悪の場合 $O(N(\log N)^2)$ である．

証明 考えている分割統治型算法において母点数が N のときの最悪の計算量を $T(N)$ と書くと，補題 4.5 により $T(N)=2T(N/2)+O(N \log N)$ が成立する (文献5)も参照のこと．最悪の場合には Katajainen らの改良が効果を生じない)．これを解くと $T(N)=O(N(\log N)^2)$ を得る． □

計算が正しいと仮定すればさらに良い値が得られる．

補題 4.6[1] 単純平面グラフ $G=(V,E)$ の頂点数を n とするとき，

$$\sum_{(v,w)\in E} \min\{\deg(v), \deg(w)\} = O(n)$$

である．

補題 4.7 性質 (C1.1)～(C1.5) を満たす GS 網 F と S を手続き HorizontalMerge に与えるとき，計算が正しければ，F の母点数と S の母点数との和を n とするときの論理関数 Parallel の評価に要する手間の総計は，$O(n)$ で抑えられる．

証明 計算が正しいときには winner の値が調整によって変わることはないから考えている手間を $C(n)$ と書き，マージ終了後の GS 網 M をグラフ G とみなして補題 4.6 を使えば，

$$C(n) \le \sum_{(v,w):\text{F-Sの枝}} 2\min\{\log(\deg(v)), \log(\deg(w))\}$$
$$\le \sum_{(v,w)\in E} 2\min\{\deg(v), \deg(w)\} = O(n)$$

である． □

定理 4.4 計算が正しいならば，与える母点数を N とするときの考えている分割統治型算法の数値的安定化後の計算量は平均的に $O(N)$，最悪の場合 $O(N \log N)$ である．

証明 まず最悪の場合については，補題 4.4 のかわりに補題 4.7 を用いて補題 4.5 から定理 4.3 に至る道筋をたどれば $O(N \log N)$ が得られる．次に平均的な場合を考える．考えている分割統治型算法において，共通接線の接点候補を1回更新するときの手間が定数であり，枝を1本削るときの手間が同じく定数であることは明らかである．さらに，一般に Delaunay 網においてある母点から出ている枝の数は平均6本であるから (例えば文献4)を見よ)，論理関数 Parallel を1回評価するのに要する手間も定数であり，これより枝を1本付加するときの手間も Parallel の評価の手間を含めて定数となる．後は文献5) の Theorem 3.2 の証明に従えば $O(N)$ が得られる． □

定理 4.4 を導くだけなら，Parallel の評価の際に平衡木を用いる必要はない．

5. 実　　験

第3章で述べた安定化された算法に基づいてプログラム (C言語で合計約 2,700 行) を作成し，数多くの計算実験を行った．その結果どんなに低い精度で計算しても決して暴走することなく計算結果を出力して終了し，計算精度を上げるにつれてその出力結果が正しい Voronoi 図に収束していくことが確認された．また計算量の面でも期待された良い性質を持つことが確認された．なお実験はすべて東京大学教育用計算機センター富士通 FACOM 380 の UNIX System V 上で行った．

図8は本プログラムの計算時間の実測結果をまとめたものである．領域 $[0,1)\times[0,1)$ 中にランダムに分布する 2^7 個，2^8 個，…，2^{15} 個の母点に対して各母点数で10回ずつ測定して平均をとった．横軸は母点数の対数を表し，縦軸は母点1個当りの平均時間を表す．本プログラムは，母点集合の分割のしかたを変えるこ

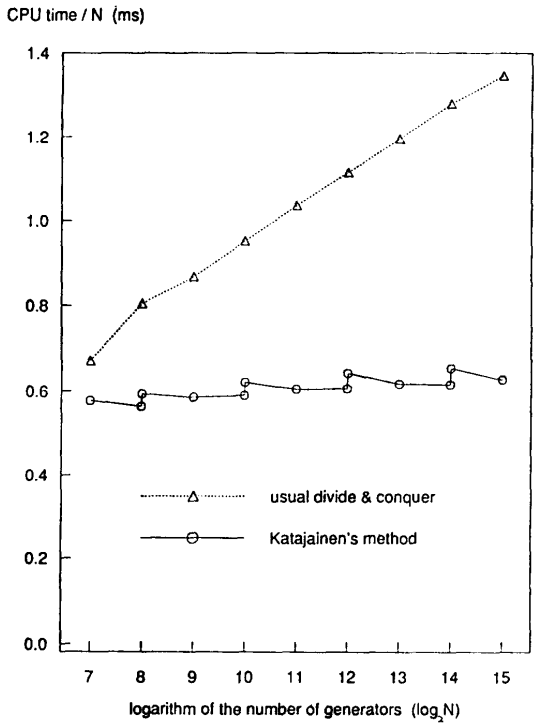


図 8 計算時間の比較
Fig. 8 Time complexity of the program.

とにより Katajainen らによる改良型の分割統治型算法だけでなく，従来型の算法で Voronoi 図構成を行うこともできる．図によってどちらの方法を用いた場合にも数値的安定化によって平均的な計算量が悪化していないことがわかる．すなわち従来型で $O(N \log N)$，改良型でほぼ $O(N)$ が達成されている．

図 9 は計算精度を上げるにつれて，プログラムの出力結果が正しい Voronoi 図に「収束」していく様子を示したものである．与えた母点は $[0,1)\times[0,1)$ 中に一様乱数を用いて発生させた 30 点である．ここで低精度計算は，浮動小数点計算を構成する一つ一つの二項演算が行われることに，その結果の仮数を 2 進数で表現したものを 0 捨 1 入によって指定された桁数に丸めることで行った．ちなみに仮数表現桁数 2 進 24 桁の精度で行った計算が IEEE の単精度の規格に一致する．

図 10 は $[0,1)\times[0,1)$ 中にランダムに分布する 40,000 点に対して単精度計算（仮数表現桁数 24 桁）で求めた Voronoi 図の左上の四分の一である．正しく構成できているように見える．かつて単精度で構成できる限界が 1,000 点だったのと比べると格段の性能の向上であり，しかも感触ではさらに多くの点に対して正しく構成できそうである（この母点数までしか実験しなかったのは 1 ユーザ当りの記憶容量制限のためである）．また同一円周上に分布する 2,000 点についても単精度計算で Voronoi 図を構成した．このような母点配置は非常に Voronoi 図を構成しにくい例として知られているが，本プログラムは破綻することなく終了し，通常の倍率で見る限り正しい Voronoi 図（正しい Voronoi 図の近似図形）を出力した[8]．

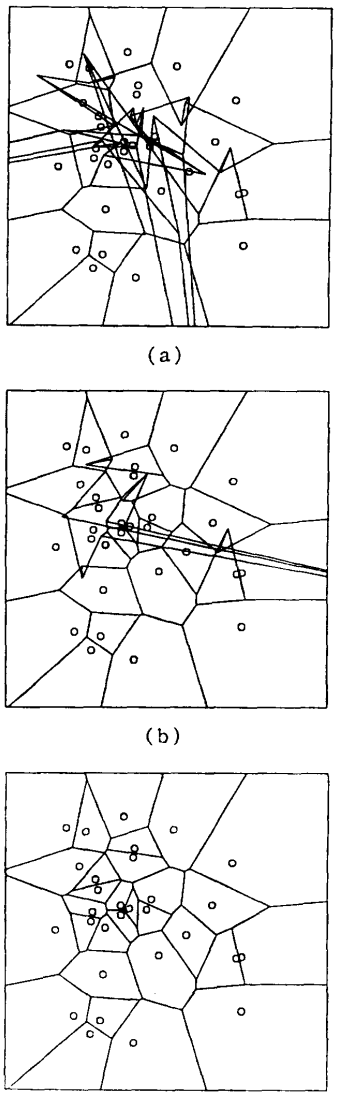


図 9 精度上昇に伴う「収束」の様子
（a） 2 進 2 桁，（b） 2 進 4 桁，（c） 2 進 6 桁
Fig. 9 Comparison of the output in different precision.
(a) in two-bit precision,
(b) in four-bit precision,
(c) in six-bit precision.

6. おわりに

最速の Voronoi 図構成算法となる可能性を持つようになった分割統治型算法を杉原・伊理の言う意味で数値的に安定にした．作成したプログラムは，期待された様々な良い性質を満足し十分実用になることがわかった．また本論文では数値的安定化とは条件 (Q1)～(Q3) を満足するように位相的歯止めをかけることと定義したが，この解釈により他の幾何的アルゴリズムへの応用も考えやすくなったと思われ，この方面の研究に一つの方向を示すことができた．

謝辞 有益な御討論を頂いた東京大学工学部計数工学科伊理正夫教授，室田一雄助教授，今井敏行助手，同理学部情報科学科今井浩助教授，機械技術研究所阿久津達也氏に感謝します．本研究は文部省科学研究費補助金（一般研究(C)，課題番号 01550279）の援助を受けている．

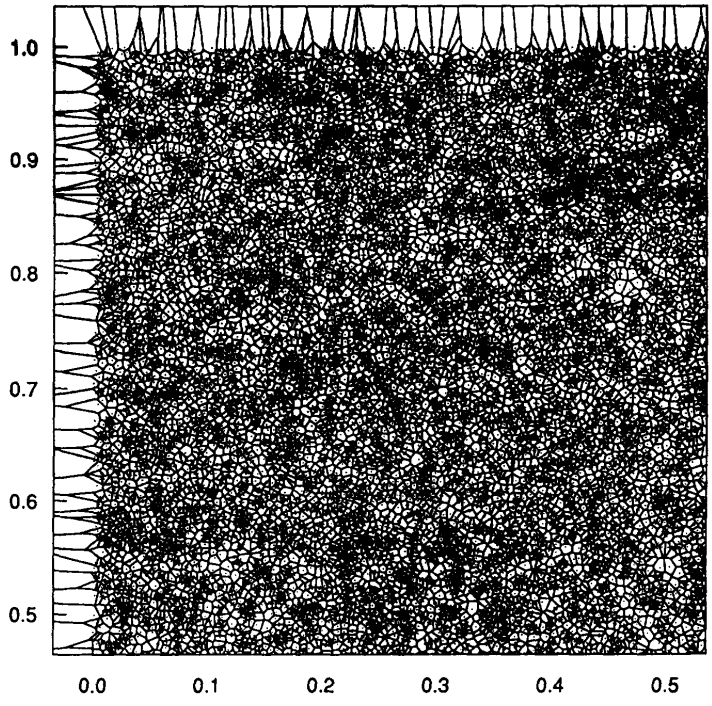
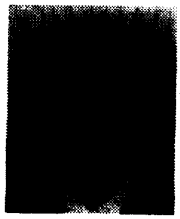


図 10 領域 $[0,1)\times[0,1)$ 中にランダムに分布する 40,000 点について単精度計算で求めた Voronoi 図（ただしその左上の四分の一．図中の黒点は母点を表す）

Fig. 10 Output of the program computed in single precision for 40,000 generators located at random in $[0,1)\times[0,1)$ (the left-upper quadrant of the whole diagram is shown; the dots represent the generators).

参考文献

1) 阿久津：平面的グラフの次数の和に関する考察，東京大学工学部計数工学科伊理研合同輪講資料(1991年2月14日)．
2) Green, P. J. and Sibson, R.: Computing Dirichlet Tessellations in the Plane, *Comput. J.*, Vol. 21, pp. 168-173 (1978).
3) Guibas, L. and Stolfi, J.: Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams, *ACM Trans. Gr.*, Vol. 4, pp. 74-123 (1985).
4) 伊理（監），腰塚（編）ほか：計算幾何学と地理情報処理，共立出版，東京 (1986).
5) Katajainen, J. and Koppinen, M.: Constructing Delaunay Triangulations by Merging Buckets in Quadtree Order, *Fundamenta Informaticae*, Vol. 11, pp. 275-288, North-Holland (1988).
6) Lee, D. T. and Schachter, B. J.: Two Algorithms for Constructing a Delaunay Triangulation, *Int. J. Comput. Inf. Sci.*, Vol. 9, pp. 219-242 (1980).
7) Ohya, T., Iri, M. and Murota, K.: Improvements of the Incremental Method for the Voronoi Diagram with Computational Comparison of Various Algorithms, *Journal of the Operations Research Society of Japan*, Vol. 27, pp. 306-336 (1984).
8) 大石：分割統治型 Voronoi 図構成算法の数値的安定化，東京大学工学部計数工学科卒業論文 (1990).
9) 大石，杉原：Voronoi 図構成における分割統治型算法の誤差対策，情報処理学会アルゴリズム研究会研究報告，90-AL-16-7 (1990).
10) Rhynsburger, D.: Analytic Delineation of Thiessen Polygons, *Geographical Analysis*, Vol. 5, pp. 133-144 (1973).
11) Shamos, M. I. and Hoey, D.: Closest-Point Problems, *Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science*, New York, pp. 151-162 (1975).
12) Sugihara, K. and Iri, M.: Construction of the Voronoi Diagram for One Million Generators in Single-Precision Arithmetic, Research Memorandum RMI 89-05, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo (1989).
13) Sugihara, K., Ooishi, Y. and Imai, T.: Topology-Oriented Approach to Robustness and Its Applications to Several Voronoi-Diagram Algorithms, Urrutia, J. (ed.), *Proceedings of the Second Canadian Conference in Computational Geometry*, Ottawa, pp. 36-39 (1990).



大石 泰章（学生会員）

1967年生．1990年東京大学工学部計数工学科卒業．現在同大学院修士課程在学中．線形システム理論，計算幾何学に興味を持つ．日本応用数理学会会員．

杉原 厚吉（正会員）

昭和23年生．昭和46年東京大学工学部計数工学科卒業．昭和48年同大学院工学系研究科計数工学専門課程修士課程修了．電子技術総合研究所研究官，名古屋大学工学部助教授等を経て，現在東京大学工学部計数工学科教授．工学博士．コンピュータビジョン，計算幾何学などの研究・教育に従事．著書 Machine Interpretation of Line Drawings (MIT Press, 1986) など．昭和55，63年度本学会論文賞受賞．電子情報通信学会，日本オペレーション・リサーチ学会，日本応用数理学会等の会員．