誤差による破綻の心配のない線分 Voronoi 図構成算法

今井敏行† 杉原厚吉†

平面上の点の勢力圏を表す図である Voronoi 図を一般化することは，理論的に興味深いだけでなく，実用上も重要である．本論文では，線分 Voronoi 図の実用的な構成算法を考案する．この新しい算法は，従来の逐次添加型の算法に，位相優先法と我々が呼んでいる数値誤差対策を適用したものである．この算法は，計算誤差による破綻を完全に防止でき，必ず結果が出力される．さらに，出力が本来持つはずの位相的な性質のいくつかが保証される．単精度浮動小数点程度の誤差が発生する環境では，線分数 n に対して，理論的に従来並みの $O(n^2)$ の速度を確保することができ，最悪の場合でも $O(n^3)$ 時間で処理を終了する．記憶量は $O(n)$ であり，理論的に最良である．また，算法を計算機上に実装し，実際には，さらに高速に計算できることも計算機実験で確かめた．

A Failure-free Algorithm for Constructing Voronoi Diagrams of Line Segments

Toshiyuki Imai† and Kokichi Sugihara†

Generalized Voronoi diagrams are not only theoretically interesting but also practically important. This paper presents a practical algorithm for constructing Voronoi diagrams of line segments. This algorithm is basically on incremental method but is modified by our new approach, called a topology-oriented approach, to a numerically robust algorithm. It is free from failure which otherwise comes from errors of computation. It always terminates normally with an output, and the output is guaranteed to have part of topological properties which the true solution should have. In ordinal levels of errors, the algorithm runs in $O(n^2)$ time, which is no slower than conventional incremental methods. The time complexity is $O(n^3)$ even in the worst case. The memory is $O(n)$, which is theoretically optimal. Experiments on computers show that the practical performance of the algorithm was better than the above theoretical complexity.

1. はじめに

平面上に有限個の点を与えたとき，それらの点による平面の勢力圏分割を表す図を Voronoi 図という．計算幾何学の理論において，当初から，Voronoi 図は中心的な研究対象のひとつであった[1),14)]．さらに，実用面においても，Voronoi 図は画像処理，数値解析から物理学，生態学や都市工学に至るまで，幅広い応用をもつ[9),13)]．

Voronoi 図を拡張して，点以外の図形による平面の勢力圏分割や3次元以上の空間の勢力圏分割を求めることは，理論面の興味に加え，実用面からの要求も強く，さらに幅広い応用が期待できる．実際，拡張された Voronoi 図の構成算法が，線分[11)]，円弧[19)]，多角形[15)]などの図形に対して提案されている．

しかし，これらの算法は一計算幾何学の研究一般と同じく一数値誤差は生じないという仮定のもとで，その正当性が保証されているに過ぎない．不動小数点計算などの誤差の発生する計算でこれらの算法を実行すると，対象の位相的構造の判定を誤り，矛盾が発生して，算法が破綻することがある．したがって，これらの算法を現実の計算環境で正常に動作するものに作りかえることは，応用上，非常に重要である．

そのための方法には，判定を正しく行うために必要な計算精度を確保する方法[20)]，誤差解析に基づいて，計算結果を信頼できるものとできないものに分離して，前者だけを利用しようとする方法[4),5),12)]，位相構造の一貫性を保つことによって矛盾の発生を防止する方法[16),17)]などがある．

本論文では，線分に対する平面の勢力圏分割を計算するための，誤差が発生しても破綻する心配のない算法を構成する．そのために，上に示した方法のうち，

† 東京大学工学部計数工学科
Department of Mathematical Engineering and Information Physics, Faculty of Engineering, the University of Tokyo

第3のものを利用する．なぜなら，この問題では曲線を扱うために，第1の方法では，必要な計算精度が実用の限界を越えてしまうし，第2の方法では，誤差解析が複雑で，誤差の大きさのタイトな見積りを得ることが難しいからである．一方，第3の方法は，このような問題にも適用できる強力な方法であり，既に，点に対する勢力圏分割などに適用され，成功を収めている[7),17)]．

第2章で問題を定義し，第3章で従来の方法と，その誤差に対する脆弱性を示す．第4章で新しい算法を構成し，第5章でその中の数値計算上の工夫を述べ，第6章で数値実験の結果を述べる．

2. Voronoi 図

$g_1, \cdots, g_n$ を，その閉包が互いに共有点を持たない平面 $\boldsymbol{R}^2$ 上の図形（点，線分など）とし，$G=\{g_1, \cdots, g_n\}$ とおく．平面 $\boldsymbol{R}^2$ 上の点 v と生成元 g_i の距離 d を，$d(v, g_i)=\inf\{\|u-v\| \mid u\in g_i\}$ で定める．ただし $\|\bullet\|$ は Euclid ノルムを表す．

曲線 $B(g_i, g_j)$ と領域 $R(g_i, g_j)$ を次のように定める：

$$B(g_i, g_j)=\{v\in\boldsymbol{R}^2 \mid d(v, g_i)=d(v, g_j)\},$$
$$R(g_i, g_j)=\{v\in\boldsymbol{R}^2 \mid d(v, g_i)<d(v, g_j)\}.$$

$B(g_i, g_j)$ を g_i, g_j の **2等分線**とよぶ．$R(g_i)=\cap_{j\neq i} R(g_i, g_j)$ とおく．$R(g_1), \cdots, R(g_n)$ が定める平面の分割を G に対する **Voronoi 図**といい，$V(G)$ で表す．G の元を $V(G)$ の**生成元**といい，$R(g_i)$ を生成元 g_i の **Voronoi 領域**という．$R(g_i)$ の境界を $\partial R(g_i)$ で表す．2個の Voronoi 領域の共通境界である曲線分を **Voronoi 辺**とよぶ．また，3個以上の Voronoi 領域に囲まれた境界上の点を **Voronoi 点**とよぶ．Voronoi 点と Voronoi 辺はグラフ的な構造をもつ．記述を簡潔にするため，グラフの用語を用いて，Voronoi 領域，Voronoi 辺，Voronoi 点を，単に，**領域**，**辺**，**節点**ともよぶことにする．

辺 b の両側が g_i, g_j の領域で，g_i, g_j, g_k の領域が節点 v を囲むとする．このとき次の性質が成り立つ．

性質 1　$b\subset B(g_i, g_j)$.

性質 2　$v\in B(g_i, g_j)\cap B(g_j, g_k)\cap B(g_k, g_i)$.

特に，点による勢力圏分割を**点 Voronoi 図**，直線分によるものを**線分 Voronoi 図**という．

線分 Voronoi 図では，辺は，直線分と放物線分が端点で接してできる複雑な曲線であり，計算機内で Voronoi 図を扱うときには，より簡単な構造が求められる．そこで，生成元を，両端点とその間の部分に3分割する．g_i が直線分で，その両端点が p_j, p_k であるとき，$e_i=g_i-\{p_j, p_k\}$ を**開線分**とよぶ．このとき，

$$B(p_j, e_i)=\{x\in\boldsymbol{R}^2 \mid (x-p_j)\cdot(p_k-p_j)=0\},$$
$$R(p_j, e_i)=\{x\in\boldsymbol{R}^2 \mid (x-p_j)\cdot(p_k-p_j)<0\},$$
$$R(e_i, p_j)=\{x\in\boldsymbol{R}^2 \mid (x-p_j)\cdot(p_k-p_j)>0\}$$

とすることによって，端点と開線分を生成元とする Voronoi 図を定めることができる．この Voronoi 図を**分割線分 Voronoi 図**とよぶことにする（**図 1**）．

生成元の分割により，辺はすべて，直線分あるいは

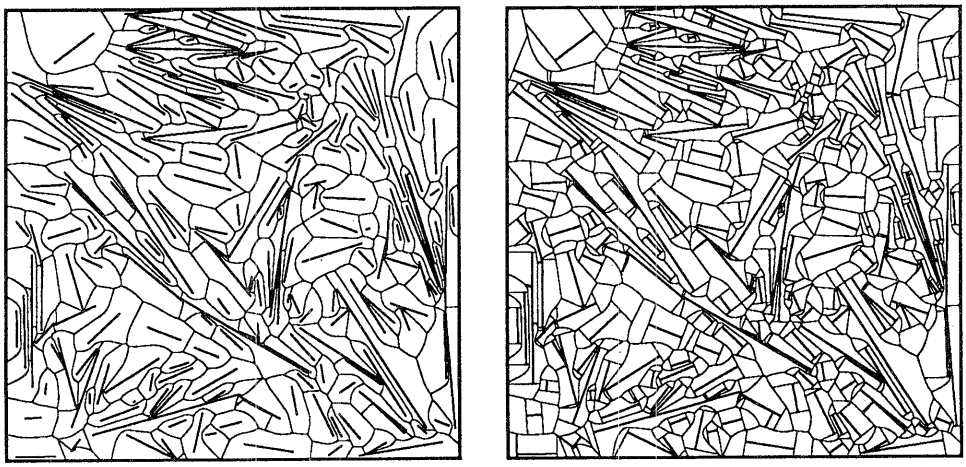


図 1　線分 Voronoi 図（a）と分割線分 Voronoi 図（b）（線分数 200）
Fig. 1　Voronoi diagrams of line-segments and separated segments (200 segments).

放物線分になり，計算機内での処理が容易になる．線分 Voronoi 図は分割線分 Voronoi 図から容易に得られる．

同様にして，開線分と端点のほかに，いくつかの点を生成元に加えても，Voronoi 図を定義できる．この Voronoi 図を**点線分 Voronoi 図**とよぶ．

本論文の算法が扱う問題を次のように設定する：

問題：単位正方形 $[0,1]^2$ 内に与えられた n 本の直線分の分割線分 Voronoi 図をこの正方形内にかけ．

適当な平行移動と拡大縮小を考えれば，この問題設定で一般性を失わない．また，次の生成元条件を仮定する．

生成元条件：生成元の直線分は，互いに共有点を持たない．

Voronoi 図の構造とグラフとの違いは，前者には無限にのびる辺が存在することである．本論文のプログラムでは，正方形 $[-1,2]^2$ を内部に含む三角形の3頂点を生成元に加えることで，無限にのびる辺を3個に限定している．このような3点を生成元に加えても，単位正方形 $[0,1]^2$ の内部では Voronoi 図に変化は起きない．

3. 構成算法と数値的不安定性

本論文で採用した点線分 Voronoi 図の構成算法は逐次添加法とよばれる算法に属す．その基本構造は次のとおりである．ただし，与えられた n 個の線分を $g_1,\cdots,g_n$ とし，g_i から端点を除いた開線分を e_i，g_i の端点を p_{2i+2}, p_{2i+3} と名付ける．さらに，前章で付加した3頂点を p_1, p_2, p_3 と名付ける．$G_0=\{p_1,\cdots,p_{2n+3}\}$ とおき，$G_i=G_0\cup\{e_1,\cdots,e_i\}$ とする．

算法 1（逐次添加法）

1. 点 Voronoi 図 $V(G_0)$ を作る．

2. $i=1,\cdots,n$ に対して **2.1** を行う．

2.1 開線分 e_i を添加し，$V(G_{i-1})$ を $V(G_i)$ に変更する．

点 Voronoi 図 $V(G_0)$ の構成には既存の算法[17]を利用するので，本論文で示す算法の本体は，開線分 e_i の添加に伴う Voronoi 図の変更である．グラフ的には，これは $V(G_{i-1})$ のグラフに，新しい領域 $R(e_i)$ の周となる辺と節点を付け加え，この領域内の古い辺と節点を消去することにより $V(G_i)$ のグラフを得ることである．

算法1の **2.1** を細かく見ると，次のようになる．

算法 2（開線分 e_i の添加）（図2）

1. 端点領域の周 $\partial R(p_{2i+2})$ と $B(p_{2i+2}, e_i)$ との交点を見付け，それを v_∞，v_0 に格納する．ただし，v_∞, v_0 の選び方は，$B(p_{2i+2}, e_i)$ に v_∞ から v_0 への向きを付けたとき，領域 $R(p_{2i+2}, e_i)$ が左になるようにする．

2. v_0 で $B(p_{2i+2}, e_i)$ と交わる辺を b_0 に格納する．

3. $g\leftarrow p_{2i+2}$，$j\leftarrow 0$．

4. $v_j\neq v_\infty$ のうちは **4.1-4.3** を行う．

4.1. b_j の両側の領域の生成元のうち，g でないものを，改めて g とする．

4.2. $\partial R(g)$ の辺を b_j から反時計回りに探索し，$B(g, e_i)$ と，はじめて交わる辺と交点を，b_{j+1}, v_{j+1} に格納する．

4.3. $j\leftarrow j+1$．

5. $v_0,\cdots,v_\infty$ を順に新しい辺で結び，最後に v_∞, v_0 を辺で結んで，$V(G_i)$ の新しい領域 $R(e_i)$ を確定する．

6. 領域 $R(e_i)$ 内の辺と節点をすべて消去する．

算法2の途中で，$\partial R(e_i)$ が $V(G_{i-1})$ の節点を通過する時，生成元の集合が退化しているという．算法2で，交点 v_∞ が $V(G_{i-1})$ の節点と一致する場合は，退化の一例である．この場合は，数値誤差を伴う計算環境でこの算法を実行すると，**1** で，v_0，v_∞ のどちらかが見付からなかったり，本来 $v_\infty=v_m$ で **4** を終了するのに，v_∞ と v_m を通る辺としてそれぞれ別の辺が指定されたりして，無限ループに陥る可能性がある．またそのために，データ構造が破壊され，開線分 e_i の添加

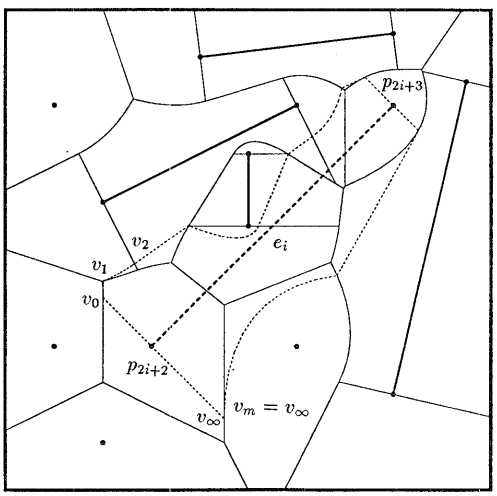


図 2 開線分の添加
Fig. 2 Adding an open segment.

前に，$R(p_{2i+2})$ が消滅し，算法2の実行が破綻することもある．これは，開線分の添加順序を変えるなどの対症療法では，必ずしも回避できない問題である．

従来，計算幾何学で提案されてきた各種算法では，非退化，無誤差計算を前提に効率化のみが論じられてきたため，通常は計算機に実装しても動作が保証されない．

4. 位相優先法

算法を計算機に実装した時，動作が保証されない理由は，計算機内で図形のデータ構造が扱えなくなるような誤判定が，計算誤差や退化によって，算法中の各所で発生するからである．

算法の動作を完全に保証するためには，次の2条件を満たせばよい．

条件 1. すべての数値計算を誤差なく行う．

条件 2. 退化に対するすべての例外処理を算法中に組み込む．

一般に，この条件1，2を満足するプログラムを書くことはプログラマに非常な困難を強いる．これを解決するため，すべての位相的な判定を整数計算で行うことによって，条件1を達成する方法[10),16),18)]，条件2を統一的な処理で達成する方法など[2),3)]の提案もある．

本論文では，これらの方法を採用せず，位相優先法とよばれる方法を採り，計算誤差を前提に，暴走やデータ破壊を回避する．図形の性質を，座標値や角度などにかかわる計量的な部分と，辺や面の接続関係などの位相的な部分に分けると，暴走やデータ破壊を導くのは位相的な部分の不合理である．そこで，図形が持つ位相的な性質を選び出し，これに反する計算結果は誤差によるものと考えて排除する．すなわち，図形の位相的な性質の無矛盾性を，数値計算結果より優先的に扱うという算法の作成法が位相優先法である．位相優先法は，算法の正常終了と，出力が本来持ついくつかの位相的性質を保証し，既存の算法とも両立が可能という柔軟性を持つ．

開線分 e_i の添加によって，Voronoi 図を $V(G_{i-1})$ から $V(G_i)$ に変更する場合に，変化する部分の位相的構造には次のような性質がある．

性質 3. 新しい節点は，辺の消去される部分と残る部分の境である．

性質 4. 辺のうち，消去される部分の全体は木構造をなす．

性質 5. 生じる辺は消去される木を囲む閉路をなす．

性質 6. 消去される木は両端点の領域 $R(p_{2i+2})$, $R(p_{2i+3})$ をつなぐ1個の道を含む．

これらは性質6を除いて，算法1の **1** で $V(G_0)$ の構成にも用いられている[17)]．文献17) ではさらに，性質6の代わりに，

性質 7. ひとつの領域の周から消去されると節点は連結である

を用いているが，これは開線分の添加時には成立しない（図2）．

上記の性質を利用して，開線分 e_i の添加の算法を次のように改良できる．

算法 3（位相優先法による開線分の添加）

1. 領域 $V(p_{2i+2})$ の周上の節点から，道の出発点を決定し，登録する．

2. 出発点から，領域 $V(p_{2i+3})$ の周に至るまで，消去される辺や節点を探索して道を見付けていき，道上の辺と節点を，順次，辺スタック B と節点スタック V に積む．

3. スタック B, V から辺や点を取り出し，それに接続する辺や節点のうち，未探索のものを探索し，消去される節点や辺があれば，スタック V, B に積むことを，スタック B, V が空になるまで行い，消去される木を確定する．

4. 消去される木を囲むように節点と辺を付け加え，消去される木を消去する．

算法2では $R(e_i)$ の境界をまず見付けて，そのあとで内部を取り除いた．一方，この算法3では，取り除くべき木をまず見付け，そのあとでそれを囲む境界を生成する．この違いが位相優先法を実現するひとつの鍵となっている．この更新算法3により，点線分 Voronoi 図はすべての節点が次数3をもつ平面グラフとして更新される．退化状態での次数が4以上の節点は，次数が3の複数の節点が，長さが0の辺で隣接しているとして扱う．次にこの算法における，出発点の決定法，道の探索法，木の確定法について順に述べる．

ここで，算法を記述するのに必要な用語をいくつか定めておく．

帯状の領域 $R(e_i, p_{2i+2}) \cap R(e_i, p_{2i+3})$ を**開線分 e_i で定まる帯状領域**とよぶ．新たに生じる領域 $R(e_i)$ はこの帯状領域の内部にある．

生成元 g_j と点 $v \in \boldsymbol{R}^2$ に対して，点 $u \in \mathrm{cl} g_j$ が $\|v-u\| = d(v, g_j)$ を満たすとき u を点 v の生成元 g_j への**射影点**という．ただし cl• は閉包を表す．点線分

Voronoi 図では，射影点は (v, g_j) の組に対し，一通りに定まるので，$\mathrm{pr}_v(g_j)$ と書くことにする．

開線分 e_i の端点 p_{2i+2}, p_{2i+3} の座標を (x_{2i+2}, y_{2i+2}), (x_{2i+3}, y_{2i+3}) とする．

また，開線分 e_i に p_{2i+2} から p_{2i+3} に向かう向きを付ける．この帯状領域内の生成元は**添加開線分 e_i に対して右側**および**左側**に分類される．

任意の節点 v に対して，v を囲む領域の生成元のひとつを g とする．このとき，節点 v に関する生成元 g の**左度** $L_v(g)$ を，射影点 $\mathrm{pr}_v(g)=(x, y)$ を用いて

$$L_v(g)=\begin{vmatrix} x & x_{2i+2} & x_{2i+3} \\ y & y_{2i+2} & y_{2i+3} \\ 1 & 1 & 1 \end{vmatrix}$$

で定める．座標系を右手系にとれば，$L_v(g)$ が正，負のとき射影点 $\mathrm{pr}_v(g)$ が，開線分 e_i を延長した有向直線のそれぞれ左側，右側にある．$L_v(g)$ が正，負のとき，生成元 g は**節点 v から見て**（開線分 e_i の）それぞれ**左，右にある**ということにする．

v と $\mathrm{pr}_v(g)$ がともに e_i で定まる帯状領域内にあれば，g が e_i に対して左右のどちら側に分類されるかと，g は v から見て e_i の左右のどちらにあるかは一致する．

Voronoi 図 $V(G_{i-1})$ で両端点の領域 $R(p_{2i+2})$, $R(p_{2i+3})$ が，ある辺を境界に隣接していれば，道としてこの辺をとればよい．隣接していない場合，道は，開線分 e_i に対して左右に分類される生成元の領域の間を通る．以下，この場合を考える．

道上の節点を出発点から道に沿って $v_0, v_1, \cdots$ とし，同様に，道上の辺を道に沿って $b_0, b_1, \cdots$ とする．b_j に v_j を始点とする向きを付けたとき，b_j の左右の領域の生成元を，それぞれ，$g_L(v_j, b_j)$, $g_R(v_j, b_j)$ とする．また，領域 $R(p_{2i+2})$ の周上の節点 v に対し，v に接続する3辺のうち，周上にない（唯一の）ものを $b(v)$ と書くことにする．道の出発点 v_0，出発辺 b_0 は次の特徴により決まる．

特徴 1. v_0 は領域 $R(p_{2i+2})$ の周上の節点である，

特徴 2. v_0 は e_i で定まる帯状領域内にある，

特徴 3. v_0 から見て $g_L(v_0, b_0)$, $g_R(v_0, b_0)$ は開線分 e_i に対しても，それぞれ左，右にある

誤差が発生する時には，数値計算に基づく判定を完全には信用できない．そこで，出発点 v_0 を次のように決定する．

算法 4（出発点の決定）（図 3）

1. 領域 $R(p_{2i+2})$ の周上の節点で，直線 $B(p_{2i+2}, e_i)$ から端点 p_{2i+3} の側に最も離れているものを v に格納する．

2. $g_L(v, b(v))$, $g_R(v, b(v))$ は（v から見て e_i に対して），それぞれ左，右にあるか，あるいは，以下の探索が領域 $R(p_{2i+2})$ の周を一周するまで調べる．

$g_L(v, b(v))$ が左ならば

時計回りに領域 $R(p_{2i+2})$ の周をみて v に隣接する点を，改めて v に格納する．

（そうでなく）$g_R(v, b(v))$ が右ならば

反時計回りに領域 $R(p_{2i+2})$ の周をみて v に隣接する点を，改めて v に格納する．

3. $v, b(v)$ を，それぞれ v_0, b_0 として終了する

探索を**1**のような節点から始めた理由は，出発点の特徴3は，帯状領域内でしか成立しないことと，帯状領域内にある領域 $R(p_{2i+2})$ の周上の節点は，最少の場合，1個しかないからである．

出発点 v_0，出発辺 b_0 から出発して，領域 $R(p_{2i+3})$ に達する消去される道の探索法を考える．

節点 v に辺 b が接続しているとする．節点 v に接続する辺を反時計回りに並べて b, $b_R(b, v)$, $b_L(b, v)$ とする．辺 $b_R(b, v)$, $b_L(b, v)$ は b, v から一意に定まる．また，$g_N(b, v)=g_R(v, b_L(b, v))(=g_L(v, b_R(b, v)))$ とする．

見付けるべき道は次の特徴により決定される．

特徴 4. 道は両端点の領域 $R(p_{2i+2})$, $R(p_{2i+3})$ を結び，

特徴 5. 道とこの2領域を除くと開線分 e_i によって定まる帯状領域は2個の連結成分に分かれるが，このとき一方の成分は開線分 e_i の右側にある生成元のみを含み，他方は左側の生成元のみを含む．

これにより，v_0, b_0 から始めて，左右の生成元の領

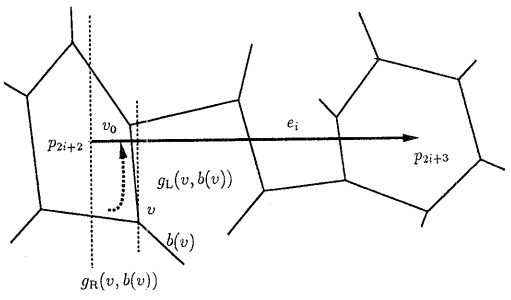


図 3 出発点の決定

Fig. 3 Deciding the starting point of the path to be removed.

域の境を，$v_1, b_1, v_2, \cdots$ と探索していけば，領域 $R(p_{2i+3})$ に達したとき，道を見付けたことになる．節点 v_{j+1} は辺 b_j から自動的に見付かり，辺 b_j は，生成元 $g_N(b_{j-1}, v_j)$ が v_j から見て左か右かに応じて，それぞれ，$b_R(b_{j-1}, v_j)$ か $b_L(b_{j-1}, v_j)$ を採ればよい．しかし計算誤差が存在するから，領域 $R(p_{2i+3})$ に探索が達する前に，無限にのびる辺に達したり，領域 $R(p_{2i+2})$ に戻ったり，既に探索された節点と辺に阻まれて，閉路を作ってしまったりする場合がある．このような状況を回避するため，これらの状況になったら，探索中に行った判定のうち最も判定が微妙なもの(すなわち，左度 $L_v(g_N(b, v))$ の絶対値が最小のもの)を選び，そこから別方向へ探索を進める (**図 4**)．すべての節点と辺が探索の対象になり得るので，数値誤差があっても，領域 $R(p_{2i+3})$ に探索が達することが保証される．データ構造として，探索が節点 v から辺 b へ進んだ時には b から v を指すポインタを持ち，逆に，辺 b から節点 v へ進んだ時には v から b を指すポインタを持てば，探索が領域 $R(p_{2i+3})$ に達した後にポインタをたどって道が容易に得られる．節点，辺が探索されたことを，フラグを立てて保持し，探索の終了後，フラグを効率よく戻すため，探索された節点，辺を積むスタック Q_v, Q_b も用意する．具体的に算法を書き下すことは容易なので省略する．以上により，道の上のすべての節点，辺を得ることができる．

消去される木の確定法を述べる．道が探索されれば，この道から順次消去される節点，辺を探索していき，消去される木を構成することができる．退化と計算誤差がなければ，辺ごとに，新しい節点が生じるか否かを判定するだけで十分であるが，計算誤差のため，生じるはずの節点を見落とし，残るはずの辺が次々と消去されると判定される可能性がある．それでも，位相優先法では，暴走することはないが，一般の数値対策と両立させることができるので，次のような基準を採用した．

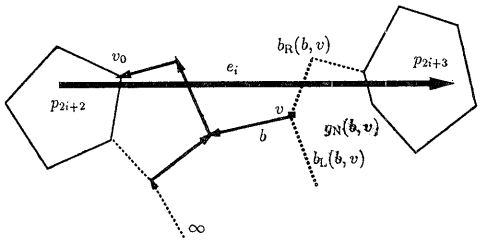


図 4　道の探索
Fig. 4　Searching the path to be removed.

基準 1.　辺に節点が生じると判定されたら必ずその判定を採用する．

基準 2.　辺に節点が生じないと判定されても，次の場合には，節点が生じるものとして扱う

判定する辺を b とし，b の両端の節点を u, v とする．ただし，v は消去されると判定済みとする．

基準 2.1.　辺 b が消去されると，消去される辺で閉路ができて木にならない，または，探索が無限にのびる辺に達する．

基準 2.2.　節点 v が，e_i で定まる帯状領域の外にある．

基準 2.3.　節点 u が，辺 b の両側の生成元よりも e_i から遠い．

基準 2.4.　節点 u が消去されると，端点領域 $R(p_{2i+2})$, $R(p_{2i+3})$ から 2 個以上の部分が消去される．

これらのうち，基準 2.1 と 2.4 が位相優先法によるもので，残りが一般の数値対策によるものである．

5. 算法の数値計算部分

位相優先法により処理が途中で破綻しないという意味の安定性が得られる．しかし，これは数値計算はいいかげんでもよいことを意味するわけでは決してない．実用的には，適切な数値計算法を選択することが，算法の性能を上げるために重要である．本論文で報告するシステムは試作段階にあり，数値計算部分に，改良の余地があると思われるが，システムの全体像を示すため，数値計算部分の概略を述べることにする．

5.1 基本計算

数値計算部分では次の基本計算を各所で用いる．

(3, 3) 行列の行列式

大型の行列ならば LU 分解などを利用する方がよいが，小型なので展開式をそのまま計算する．

2 次方程式の求解

計算誤差対策は分母の有理化[8]のみとした．また，方程式が不定になる場合は解なしとして扱う．

3 点の向き

同次座標で考える．3 点 $(X_1, Y_1)/Z_1$, $(X_2, Y_2)/Z_2$, $(X_3, Y_3)/Z_3$ で，$Z_1, Z_2, Z_3 \geq 0$ とする．

行列式 $\begin{vmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ Z_1 & Z_2 & Z_3 \end{vmatrix}$ が正の時，この 3 点は正の向きに並ぶといい，負の時は負の向きに並ぶということにする．算法中では同次座標を用いるときは，Z 座標が非負になるように各座標の符号を調整している．ま

た，$(x, y)\in \boldsymbol{R}^2$ を同次座標として扱うときには，$(x, y)/1$ とする．

2直線の交点

同次座標で考えることにし，第1の直線上の2点を $(X_1, Y_1)/Z_1, (X_2, Y_2)/Z_2$ とし，第2の直線上の2点を $(X_3, Y_3)/Z_3, (X_4, Y_4)/Z_4$ とする．この2直線の交点 $(X, Y)/Z$ は，

$$\begin{bmatrix}X\\Y\\Z\end{bmatrix}\leftarrow\begin{vmatrix}X_2 & X_3 & X_4\\Y_2 & Y_3 & Y_4\\Z_2 & Z_3 & Z_4\end{vmatrix}\begin{bmatrix}X_1\\Y_1\\Z_1\end{bmatrix}-\begin{vmatrix}X_3 & X_4 & X_1\\Y_3 & Y_4 & Y_1\\Z_3 & Z_4 & Z_1\end{vmatrix}\begin{bmatrix}X_2\\Y_2\\Z_2\end{bmatrix}$$

で求める．ただし，$Z\geq 0$ となるように，必要があれば符号を一斉に変える．この記法により無限遠点を例外扱いする必要がなくなり，すべての場合を統一的に扱える．

5.2 Voronoi 点の計算

節点は，どのような生成元の領域で囲まれるかによって，次の6個の型に分類できる．

(点，点，点) 型：3個の点，

(点，端-線) 型：1個の開線分，2点，そのうちただ1個がその開線分の端点，

(点，点，線) 型：1個の開線分，その端点でない2点，

(端-線，線) 型：2開線分，そのただ一方の端点である1個の点，

(点，線，線) 型：2開線分，その端点でない1個の点，

(線，線，線) 型：3個の開線分．

求める節点の候補がどの型に入るかは，組合せ構造からわかるので，数値計算は不要である．開線分 e_i の添加の時には，3つの生成元のうち1つは e_i になるので，(点，点，点) 型の節点は生じないから，この型の節点を求める手続きは設けなかった．

以下では各型の節点の座標値を求める手続きについて述べる．共通の記号として，求める節点を u，その座標を (x, y)，u を反時計回りに囲む生成元を g_i，g_j，g_k として，開線分より点を先行して書く．それでも任意性が残る場合，i，j，k のうち，最小のものが i になるようにして，g_i，g_j，g_k の並び方を一意に定める．また，本節では，局所的な変数として，点の名前 p_l とその座標値 (x_l, y_l) $(l=1, \cdots, 6)$ を用いることにする．

u の座標値は，2等分線の交点として求められ，生成元 g_i, g_j, g_k の位置から直接に計算する．これは，誤差が累積するのを防ぐためである．また，u を囲む生成元の順序が特定されているので，放物線と直線の交点のように複数の交点がある場合でも，不要な方の交点の座標値を求めない．これによって，不要な交点の座標値によるオーバーフローを回避できる．

5.2.1 (点，端-線) 型の場合

g_i，g_j は点で，それぞれ p_1，p_2 とする．g_k は開線分で p_1 または p_2 は端点である．反対側の端点を p_3 とする．

g_k が p_2 に接続する場合

p_1, p_2, p_3 が正の向きに並ぶならば，直線 $B(p_1, p_2)$，$B(p_2, g_k)$ の唯一の交点が u である．

g_k が p_1 に接続する場合

前の場合の p_1，p_2 が入れ替わる．

いずれの場合も，3点が正の向きに並ばない時は，交点なしとして扱う．

5.2.2 (点，点，線)，(端-線，線)，(点，線，線) の各型の場合

これらの各型の節点は，放物線と直線の交点として得ることができる．2次方程式の解の公式により，交点の座標値が解析解として求まるので，反復法などによらず，直接に求める．放物線である2等分線は点の生成元を焦点，開線分を準線の一部にする．そこで，各座標値に，平行移動と回転を組み合わせた変換 $(x, y)\mapsto(x', y')$ を施して，準線が x' 軸に平行で，下に凸な放物線に変換し，変換後の放物線と直線の式から，y' を消去して x' を求める．焦点を p_1，準線を直線 p_2p_3 (ただし，p_1, p_2, p_3 が正の向きに並ぶようにする)，回転の中心 (変換後の原点) を p_4 とすると，変換は次のようになる．

$$d\leftarrow\sqrt{(x_3-x_2)^2+(y_3-y_2)^2},$$
$$c\leftarrow(x_3-x_2)/d,$$
$$s\leftarrow(y_3-y_2)/d,$$
$$\begin{bmatrix}x'\\y'\end{bmatrix}\leftarrow\begin{bmatrix}c & s\\-s & c\end{bmatrix}\begin{bmatrix}x-x_4\\y-y_4\end{bmatrix},$$
$$\begin{bmatrix}x\\y\end{bmatrix}\leftarrow\begin{bmatrix}c & -s\\s & c\end{bmatrix}\begin{bmatrix}x'\\y'\end{bmatrix}+\begin{bmatrix}x_4\\y_4\end{bmatrix}\text{ (逆変換)}.$$

(点，点，線) 型の節点の場合

g_i，g_j は点，g_k は開線分である．g_k の両端を p_2，p_3 にとる．また，p_4 を線分 g_ig_j の中点にとる．g_k の両端のどちらを p_2, p_3 にするかは，p_4, p_2, p_3 が負の向きに並ぶ時に限り，p_2, p_3 を入れ替えることにより決めた．放物線と直線の方程式から導かれる2次方程式が解なし，あるいは，$y_2'=0$ の時には，節点は生じないとした．

(端-線，線) 型の場合

g_i が点，g_j, g_k は開線分でどちらか一方のみが g_i に接続する．

g_j が g_i に接続する場合

g_i を回転の中心 p_4 にとる．開線分 g_k の両端に接続する点を p_2，p_3 とする．p_4, p_2, p_3 が負の向きに並ぶ時に限り，p_2，p_3 を入れ替える．

g_k が g_i に接続する場合

g_j と g_k の役割を入れ替える．

直線と放物線の方程式から導かれる2次方程式を解くとき，解の公式の符号の選び方は，g_j，g_k のどちらが g_i と接続するかによって変わる．2次方程式が解なし，あるいは，$y_2'=0$ の時には，節点は生じないとした．

(点，線，線) 型の節点の場合

g_i は点であり，これを回転の中心 p_4 とする．また，g_j，g_k は開線分であり，どちらの端点も g_i と一致しない．開線分 g_k の両端に接続する点を p_2，p_3 とし，g_j の両端に接続する点を p_5，p_6 とする．p_4，p_2，p_3 や p_4，p_5，p_6 が負の向きに並ぶ時に限り，それぞれ p_2，p_3 や p_5，p_6 を入れ替えることにし，p_2，p_3，p_5，p_6 の選び方を一意にした．これを利用して，開線分 g_j，g_k の2種類の2等分線のうち，どちらの直線を採るかの判定に利用した．この直線と放物線の方程式から導かれる2次方程式が解なしの時には，節点は生じないものとして扱い，$y_2'=0$ の時には g_i と g_j が近接しているとみなし $y'=0$ とした．

これら各型の節点を求めるときの，各点 p_t の選び方から，解くべき2次方程式で解の公式の符号を決定することができ，不要な点の座標値の計算を避けられる．

5.2.3 (線，線，線) 型の場合

g_i，g_j，g_k は開線分である．開線分 g_j，g_k を，必要ならば延長して，交点 q_i を同次座標で求め，$(X_i, Y_i)/Z_i$ とする．同様にして，g_k，g_i と g_i，g_j から，それぞれ，交点 q_j と q_k を同次座標で求める．この時，$Z_i, Z_j, Z_k \geq 0$ になる．節点の候補 u は，q_i，q_j，q_k が正の向きに並ぶならば，三角形 $q_iq_jq_k$ の内心で，他は傍心になる．

内心や傍心の同次座標 $(X, Y)/Z$ は次のように計算できる．

$$D_\alpha \leftarrow \sqrt{(X_\beta Z_\gamma - X_\gamma Z_\beta)^2 + (Y_\beta Z_\gamma - Y_\gamma Z_\beta)^2}$$
$$((\alpha, \beta, \gamma) = (i, j, k), (j, k, i), (k, i, j))$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \leftarrow \pm D_i \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \pm D_j \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix} \pm D_k \begin{bmatrix} X_k \\ Y_k \\ Z_k \end{bmatrix}.$$

複号は，すべて正の時は内心，第1の複号だけが負の符号をとると点 q_i に関する（傍接円が辺 q_jq_k に接する）傍心で，第2，第3の複号だけが負の時は，それぞれ点 q_j，q_k に関する傍心となる．複号の残りのとり方は上の場合のどれかに帰着できる．

q_i，q_j，q_k が正の向きに並ぶ場合には，複号をすべて正にして計算する．正の向きに並ばない場合には，求める節点 u は傍心である．開線分 g_i，g_j，g_k の相対的な位置関係を調べ，q_i，q_j，q_k のどれに関する傍心かを判定する．詳細は省略する．

$Z=0$ の時や，q_i，q_j，q_k に関する傍心の条件を満たさない時は，節点は生じないものとして扱った．

6. 算法の計算量，記憶領域

算法の計算量，記憶量について考える．生成元の総数を n とする．

理論的には，線分 Voronoi 図の構成には，$O(n)$ の記憶量と $O(n \log n)$ の計算量が必要である．これらを達成する算法に，再帰二分法[19]があるが，プログラムが複雑になるため，本論文では採用しなかった．

線分Voronoi図，点線分Voronoi図では，点Voronoi図と同様に，節点，辺の数はともに $O(n)$ である．

プログラム中で用いたデータ構造では，Voronoi 図として，辺，節点，領域ごとに，接続関係を示す定数個のポインタを持ち，また，全節点の座標値を保持する．これに必要な記憶量は $O(n)$ である．ほかには，辺や節点の探索用のスタック，フラグ，リスト，左度の格納場所などが $O(n)$，局所変数は定数個で，算法全体では $O(n)$ の記憶量である．

算法の理論的な計算量を各部分について考える．まず，点 Voronoi 図の構成に $O(n^2)$ かかる．次に，開線分1個の添加ごとに，道の出発点を決定するために辺の総数に比例する $O(n)$ の計算量，道を探索する時には，行き詰まった場合に，探索をやり直す節点を探すのに $O(n)$ かかると見積って，結局，$O(n^2)$ の計算量がかかる．道から木構造を得るのには，やはり，辺の総数である $O(n)$ の計算量がかかる．したがって，算法全体の計算量は $O(n^2+n(n+n^2+n))$ すなわち $O(n^3)$ と見積ることができる．ただし，これは数値計算の精度が非常に悪く，ほとんど無意味な振る舞いをすると仮定した最悪の場合の計算量である．

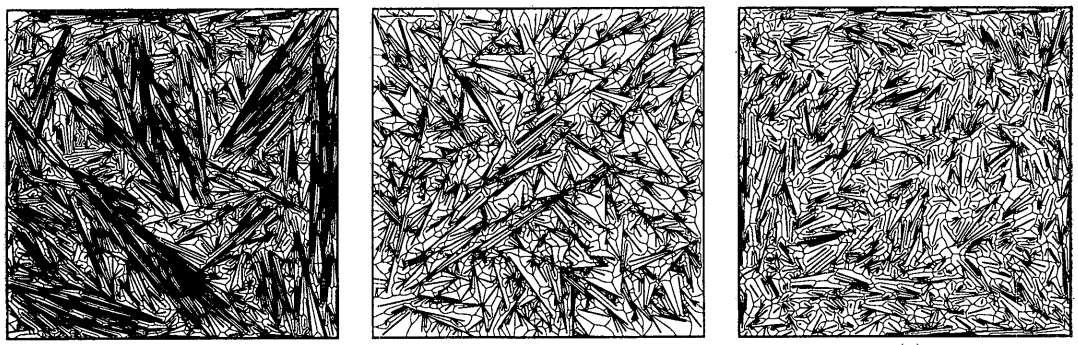


図 5 取り方1(a), 取り方2(b), 取り方3(c)による Voronoi 図の例
Fig. 5 Voronoi diagrams of various types of distribution of segments.

道を探索する時，既探索節点をヒープに積んで，探索に行き詰まる場合に，再探索節点を効率よく求めることができる．この場合，算法全体の計算量を $O(n^2 \log n)$ に減らすことができる．しかし，これでは，探索が行き詰まらなくても，算法全体の計算量は $O(n^2 \log n)$ のままになる．本論文では実用性を重視し，このような手段を採らないことにした．

計算量を各部分でみると，算法全体の計算量を決めているのは道を探索する部分である．この部分の計算量は，探索に行き詰まることがなければ，開線分1個の添加ごとに $O(n)$ ですむ．この時には，算法全体の計算量は $O(n^2)$ になる．

現実的には，道を探索する，あるいは木構造を得るために，全辺を探索することはほとんどないであろう．多くの実際的な場合に，計算量 $O(n^2)$ の見積りは過大である．実際，点 Voronoi 図の構成では，生成元が一様に分布するならば，平均計算量は $O(n)$ になる[17]．本論文で提案する算法の実際の計算時間について調べるため，実験をした．使用した計算機は富士通 S-4/EC である．

線分は両端点の順序対として与え，第1の端点を前端点，第2の端点を後端点とよんで区別する．実験の全体を通じ，端点の初期の座標は，[0, 1] の一様乱数によって与えた．入力線分の取り方は次の3種類である．

取り方 1. 線分が第 i 番目まで得られている時，第 $i+1$ 番目の線分を取り，第1番目から第 i 番目までの線分との交差判定を順に行い，交差しなくなるまで，第 $i+1$ 番目の線分を取り直すことを，第 n 番目の線分が得られるまで繰り返す．

取り方 2. 線分が第 i 番目まで得られている時，第 $i+1$ 番目の線分をまず仮に取り，第1番目から第 i 番目までの線分との交差判定を順に行い，交差する度に，第 $i+1$ 番目の線分の後端点を，交点と前端点を結んだ線分の 1：9 の内分点に置き換えることを，第 n 番目の線分が得られるまで繰り返す．

取り方 3. まず n 個の線分を仮に取り，第 i 番目と第 j 番目の線分 $(i<j)$ の交差判定を行い，交差したら，それぞれの線分の後端点を入れ換えることを，交差がなくなるまで繰り返す．第 i 番目と第 j 番目の線分の交差判定は (i, j) の辞書式順序で小さいほうから行う．

理論的には，これらの取り方で，入力線分が有限時間で得られる．線分数が 1024 本の場合の取り方1，2，3による Voronoi 図の例を**図5**に示す．

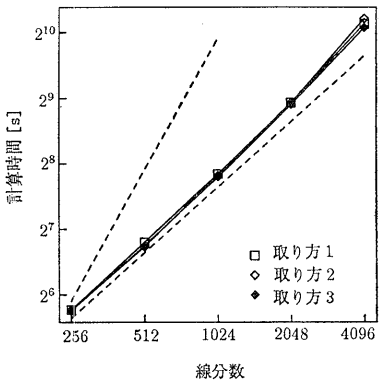


図 6 線分数と計算時間
Fig. 6 Time required for constructing Voronoi diagrams.

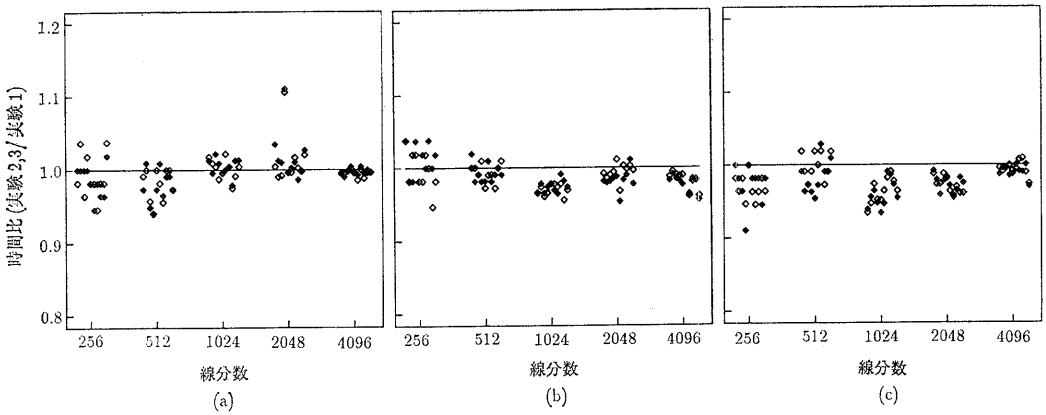


図 7 添加順序による計算時間の比較
Fig. 7 Comparison of computation times in various orders of segments to be added.

次の3種類の実験を行った.

実験 1. 取り方 1, 2, 3 によって得られた線分を生成元とする Voronoi 図を生成元数が 265, 512, 1024, 2048, 4096 の時, それぞれに対して 10 回ずつ求め, 計算時間を計測した.

実験 2. 取り方 1, 2, 3 によって得られた入力データを線分の長さの長いものから短いものへ並べ換え, 実験1と同様に時間を計測した.

実験 3. 取り方 1, 2, 3 によって得られた入力データを線分の長さの短いものから長いものへ並べ換え, 実験1と同様に時間を計測した.

表 1 実験の結果 (単位は秒)
Table 1 Total time (seconds) to construct 10 Voronoi diagrams of each number of segments.

線分数	取り方1	取り方2	取り方3
実験1			
256	54.7	54.4	54.5
512	111.5	106.6	107.7
1024	229.8	226.0	223.9
2048	488.9	487.4	481.2
4096	1119.5	1183.6	1076.0
実験2			
256	53.9	54.9	53.0
512	108.9	105.8	105.9
1024	230.8	220.1	214.7
2048	498.0	478.8	468.5
4096	1115.3	1084.2	1065.1
実験3			
256	54.2	54.2	52.8
512	109.2	105.9	107.6
1024	230.0	219.1	215.7
2048	494.8	484.1	467.3
4096	1113.8	1087.6	1069.8

実験の結果を**表1**に示す.

生成元の取り方ごとに, 10回の線分の生成で, 線分の長さに関する分布は大きな差はなかった.

次に**図6**に取り方1, 2, 3における線分数と各線分数で Voronoi 図を 10 回生成するのに要した合計時間の関係を両対数グラフで示す. 各グラフにおいて破線は計算時間の合計が線分数に比例することと, 2乗に比例することを表す傾きの直線である. グラフから, どの取り方でも線形は越えるものの, 線形に近い計算時間を持つことがわかる.

算法から, 計算時間は消去された辺の総数程度であることがわかる. 辺の総数は, 生成元の添加につれて増加する. そこで, より多くの辺を消去すると思われる長い開線分を生成元が増えないうちに処理したほうが計算時間が短くなると予想できる.

入力線分データごとに実験2, 実験3での計算時間を実験1での計算時間で割った値を, それぞれ白い菱形と黒い菱形で示したグラフが**図7**である. 菱形が縦軸の 1.0 より下にあれば, 線分の並べ換えにより, 計算時間が短縮されたことになるが, 予想した計算時間の変化は, 明確には確認できなかった. 理由としては点 Voronoi 図を構成した時点で, 全体の 2/3 の生成元が, 添加を終えているため, 開線分を添加する時には, 既に辺の総数が十分に多くなっていることなどが考えられる.

7. ま と め

線分 Voronoi 図の構成算法を, 位相優先法に基づいて改良し, そのプログラムを試作した. このプログ

ラムは，どのような数値計算誤差，退化状況の下でも，暴走や異常終了せず，正常に終了するという意味の数値的安定性をもつ．入力線分数 n に対して，最悪の計算量は $O(n^3)$ であるが，従来の方法で Voronoi 図が求められる程度の，誤差，退化状況ならば，従来どおり $O(n^2)$ の理論的上限をもつ．さらに，実際には平均的に $O(n)$ に近い計算時間で処理が終了することを実験によって確かめることができた．記憶量は $O(n)$ であり，理論的にも最良である．従来のプログラムは，終了する保証がなく，出力が得られた場合でも，その出力の品質に何ら保証がないという，実用上の難点があった．しかし，本論文のプログラムは，最低限，必ず終了し，出力が得られ，出力が平面グラフであることを保証し，精度と速度を落とさずに，計算精度に見合った数値結果を出すため，実用的である．なお，開線分の添加を途中で止めれば，点と線分の Voronoi 図も得られる．

現段階での問題点は，最悪の計算量の見積りが $O(n \log n)$ でないこと，端点の処理を初めに行っているため入力に対し静的で，生成元を追加できないこと，数値計算部分に改善の余地があることなどである．

これらの問題の解消のほかに，次のような課題が残っている．

計算量の理論面での解明：実験で測定された計算量は $O(n^2)$ と較べても小さい．これについては，算法のより詳細な分析が必要である．

テスト用の入力の生成法：実験に使用した素朴な線分生成プログラムでは，現状でも，入力データの計算に，しばしば，Voronoi 図の構成よりも多い時間がかかるうえ，計算誤差や疑似乱数の質のために計算が無限ループに陥り，入力が得られないことすらある．

生成元条件の緩和：生成元の線分が端点を共有することを，1端点に2開線分まで許すと，折れ線や単純多角形にまで，生成元を一般化できる．この，点，線分，折れ線，多角形を生成元とする Voronoi 図の構成プログラムを試作中である．また，入力で，1端点を3以上の開線分が共有する場合には，入力データの位置に関して，位相的な拘束が強く働くので，点線分 Voronoi 図に較べてまだ不明な点が多い．

幾何的アルゴリズムの多くに，位相優先法を適用し，数値的安定化を図れると思われる．その有効性を実証するため，より多くの具体的問題に，この方針を適用していきたい．また，退化現象や暴走に至る一般的な機構について，より緻密な検討の必要を感じている．

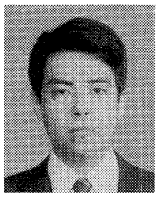
本研究の一部は科学研究費補助金（05700062, 04452191）の援助を受けている．

参考文献

1) Aurenhammer, F.: Voronoi Diagrams—A Survey of a Fundamental Geometric Data Structure, *ACM Comput. Surv.*, Vol. 23, pp. 345-405 (1991).
2) Edelsbrunner, H. and Mücke, E. P.: Simulation of Simplicity—A Technique to Cope with Degenerate Cases in Geometric Algorithms, *Proc. 4th ACM Annual Symposium on Computational Geometry*, pp. 118-133 (1988).
3) Fortune, S.: Stable Maintenance of Point-set Triangulations in Two Dimensions, *Proc. 30th IEEE Annual Symposium on Foundations of Computer Science*, pp. 494-499 (1989).
4) Guibas, L., Salesin, D. and Stolfi, J.: Epsilon Geometry—Building Robust Algorithms from Imprecise Calculations, *Proc. 5th ACM Annual Symposium on Computational Geometry*, pp. 208-217 (1989).
5) Greene, D. H. and Yao, F.: Finite-resolution Computational Geometry, *Proc. 27th IEEE Annual Symposium on Foundations of Computer Science*, pp. 143-152 (1986).
6) 今井敏行，杉原厚吉：組合せ構造を優先した線分ボロノイ図の構成法，情報処理学会研究報告，89-AL-11-2 (1989).
7) Inagaki, H., Sugihara, K. and Sugie, N.: Numerically Robust Incremental Algorithm for Constructing Three-dimensional Voronoi Diagrams, *Proc. 4th Canadian Conference on Computational Geometry*, pp. 334-339 (1992).
8) 伊理正夫：数値計算の常識，共立出版 (1985).
9) 伊理正夫（監修），腰塚武志（編集）：計算幾何学と地理情報処理，第2版，共立出版 (1993).
10) Karasick, M., Lieber, D. and Nackman, L. R.: Efficient Delaunay Triangulation Using Rational Arithmetic, *ACM Trans. Graphics*, Vol. 10, pp. 71-91 (1991).
11) 小久保岩生：一般化 Voronoi 線図の構成算法の研究—特に線分に対する Voronoi 線図について，東京大学大学院工学系研究科計数工学専門課程修士論文 (1985).
12) Milenkovic, V.: Verifiable Implementations of Geometric Algorithms Using Finite Precision Arithmetic, *Artif. Intell.*, Vol. 37, pp. 377-401 (1988).
13) Okabe, A., Boots, B. and Sugihara, K.: *Spatial*

Tessellations: Concepts and Applications of Voronoi Diagrams, John Wiley & Sons (1992).
14) Preparata, F. P. and Shamos, M. I.: *Computational Geometry*, Springer-Verlag (1985).
15) Srinvasan, V. and Nackman, L. R.: Voronoi Diagram for Multiply-connected Polygonal Domains I: Algorithm, *IBM J. Res. Dev.*, Vol. 31, pp. 361-372 (1987).
16) 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, pp. 962-974 (1987).
17) Sugihara, K. and Iri, M.: Construction of the Voronoi Diagram for "One Million" Generators in Single-precision Arithmetic, *Proc. IEEE*, Vol. 80, pp. 1471-1484 (1992).
18) Sugihara, K.: A Simple Method for Avoiding Numerical Errors and Degeneracy in Voronoi Diagram Computation, *IEICE Trans. Fundamentals*, Vol. E 75-A, pp. 468-477 (1992).
19) Yap, C. K.: An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments, *Discrete Comput. Geom.*, Vol. 2, pp. 365-393 (1987).
20) 吉田清範：代数的な量の符号判定に必要な計算精度，電子情報通信学会論文誌，Vol. J 69-A, pp. 543-547 (1986).

(平成5年10月4日受付)
(平成6年6月20日採録)



今井 敏行（正会員）

1964年生．1987年東京大学理学部数学科卒業．1989年東京大学大学院工学系研究科計数工学専攻修士課程修了．現在，東京大学工学部計数工学科助手．計算幾何学の研究に従事．日本数学会，日本応用数理学会各会員．

杉原 厚吉（正会員）

昭和46年東京大学工学部計数工学科卒業．昭和48年同大学院修士課程修了．電子技術総合研究所，名古屋大学工学部を経て，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「不可能物体の数理」（森北出版），「計算幾何工学」(培風館)，Machine Interpretation of Line Drawings (MIT Press), Spatial Tessellations —Concepts and Applications of Voronoi Diagrams (Wiley, 共著) など．日本応用数理学会等の会員．