1M-2

GPUを用いたドローネ三角形分割アルゴリズムの効率化について

河野 勇人† 山本 修身‡

名城大学大学院 理工学研究科† 名城大学 理工学部 情報工学科‡

1 はじめに

本稿では，GPUなど並列計算環境におけるドローネ三角形分割の効率的な計算アルゴリズムについて考える．ドローネ三角形分割はユークリッド距離によるボロノイ図の双対図形である．ボロノイ図は平面上に与えられた n 点（母点）によって平面上の各点が母点に一番近いかで領域分割したものであり，ドローネ三角形分割は，接した2領域に対応する2母点間に辺を付け加えたグラフ構造のことである（図1参照）．ボロノイ図やドローネ三角形分割は，地理情報システムをはじめ色々な分野に応用することができる．

ボロノイ図およびドローネ三角形分割を計算するためのアルゴリズムは多く研究されており，n 個の母点のボロノイ図を計算するのに $O(n \log n)$ の計算量で計算することができる [1]．ここで考えるユークリッド距離ばかりでなくもっと一般的な距離に関するボロノイ図を GPU を用いて計算する方法として，Hoff の方法 [2] などが研究されてきた．GPU では各画素に平面上の点を割り当て，それぞれの点がどの母点に対応する領域に含まれるかを並列に計算する．Hoff の方法の改良版として Rong のアルゴリズム [3] などが知られている．これらのアルゴリズムから得られる情報はあくまで離散した点に関するボロノイ図であり，この情報からドローネ図を計算しようとした試みとして [4] がある．しかし，離散したボロノイ図は領域の隣接情報としては，完全な情報を含まず，整合性のある隣接情報を取り出すことが困難である．本稿では，離散ボロノイ図を拡張することにより得られた情報から正確なボロノイ図を計算する概略について述べた後，ドローネ三角形を抽出する部分の最適化について述べる．

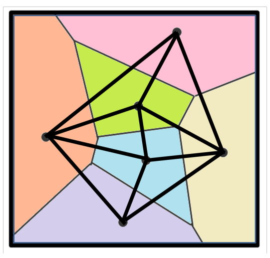

図 1: ボロノイ図とドローネ三角形分割

2 アルゴリズムの概略

ここで考えるアルゴリズムは，前述のとおり，離散ボロノイ図の拡張を計算と，その結果を用いたドローネ三角形の抽出によって構成される．ここでは前半の計算の概略を主に示す．基本的な考え方は，現在考えている領域の部分領域にボロノイ領域が入らない母点を削除していくということである．1つの大きな領域にすべての母点が入っている状態からスタートして*，図2に示すように順次領域を細分化していく．このとき，それぞれの部分領域からみた母点の順序関係（$p_1 \succ_A p_2$ ならば，領域 A に p_2 の領域は入らない）を定義し，それを手がかりに母点の削減を行う．このような処理によって，細分化されたそれぞれの領域に，その領域にボロノイ領域を作る母点の集合が列挙される．アルゴリズムの後半では，この情報から，それぞれの部分領域に外接円の中心を持つようなドローネ三角形を列挙する．そのアルゴリズムについては次節で解説する．

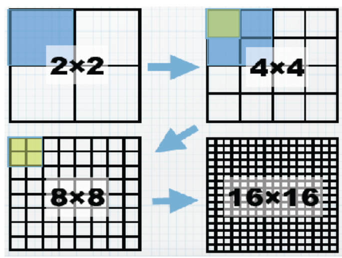

図 2: 段階的な短形分割による母点選別の流れ

*On heuristics for an algorithm for computing Delaunay triangulation using a GPU

†Yuto Kawano, Graduate School of Science and Technology, Meijo University

‡Osami Yamamoto, Department of Information Engineering, Faculty of Science and Technology, Meijo University

*説明では普通の矩形を考えているが，実際には無限に広いユークリッド平面を初期領域としなければならず，点の射影変換が必要となる．

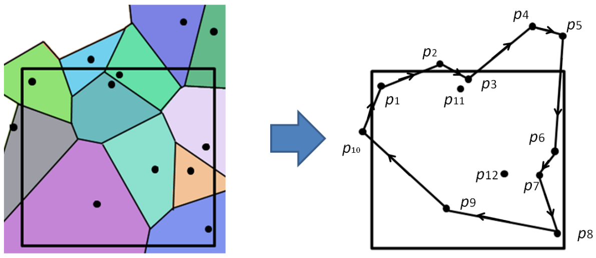

図 3: 外部点の順序関係

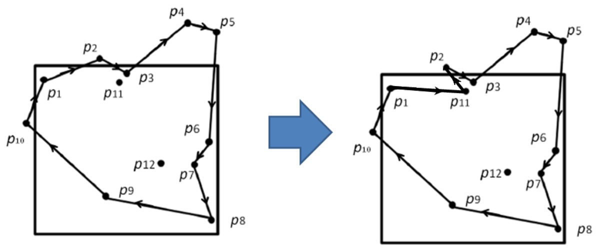

図 4: 探索による外部点更新

3 比較的少数母点での三角形探索法

区間毎に単純化された問題の探索法について述べる．それぞれの問題は，入力として区間座標と，その区間内に領域が存在する可能性を持つ母点をとる．ドローネ三角形の単純な計算法に，母点全ての 3 つの組み合わせに対し計算する方法がある．この方法では，ある区間内に n 個の母点が存在するとき，$O(n^3)$ という組み合わせから，膨大な計算回数を必要とする．そこで，効率的な計算を行うために，区間内にボロノイ領域を持たない母点を完全に除外する．これは，区間にボロノイ領域を持たない母点が「区間内に母点が存在しない．且つ，区間の枠上にボロノイ領域を持たない」という条件を満たすことを利用する．枠上に存在する全てのボロノイ領域の列挙については，枠上に複数の探索点を配置し，隣接した探索点による母点の位相構造の比較を行うことで，領域が現れる順序関係を含め，完全に列挙することができる [5]．区間の枠上にボロノイ領域をとる母点を外部点，区間内に存在するが外部点でない点を内部点と分類し，外部点の順序関係より，ドローネ三角形を探索する方法について述べる．

図 3 では，外部点が $\{p_1, p_2, p_3, ..., p_{10}\}$，内部点が $\{p_{11}, p_{12}\}$ となっている．このとき，外部点の隣接する順序関係 (p_1, p_2), (p_2, p_3) 等は，ボロノイ境界面を示す．そこで，ボロノイ境界面を持つ母点で必ずドローネ三角形が構成されることを利用し，2 点を固定させ，残り 1 点について計算を行う．ただし，ボロノイ境界面 (p_1, p_2) に注目した場合，ドローネ三角形要素となる可能性を持つものは，外部点の順序関係として前後に位置する $\{p_3, p_{10}\}$ と内部点 $\{p_{11}, p_{12}\}$ の 4 点であり，全ての母点に対し探索を行う必要はない．

さらに，(p_1, p_2, p_{11}) などドローネ三角形要素が発見された場合の処理として，その探索によって明らかになったボロノイ境界面を利用し，外部点と内部点を更新することで，常に 2 点の固定した動作をとる．この場合，外部点順序を $\{p_1, p_{11}, p_2, p_3, ...\}$，内部点を p_{12} と更新する (図 4 参照)．探索と外部点，内部点の更新を同時に行うことで，常に 2 点を固定しながら，区間の外から内へドローネ三角形要素を探索する．

10 種類のランダムな 50 点の母点に対し，この探索法を実行することで，平均探索回数が 19600 から 428.3 に減少した．また，Python 環境で平均実行時間が 0.87 秒から 0.03 秒に変化した．今後，母点と区間分割数の最適値の考察など，並列処理部の考察が必要である．

参考文献

[1] Fortune, S. J.: A sweepline algorithm for Voronoi diagrams. *Algorithmica*, vol. 2, pp. 153-174, 1987.

[2] Hoff, K. E., Culver, T., Keyser, J., Lin, M. and Manocha, D.: Fast computation of generalized Voronoi diagrams using graphics hardware. In *Proceedings of ACM SIGGRAPH annual Conference on Computer Graphics*, ACM, pp. 277-286, 1999.

[3] Rong, G., Tan, T.: Jump flooding in GPU applications to Voronoi diagram and distance transform. *The Proceedings of ACM Symposium on Interactive 3D Graphics and Games*, pp. 109-116, 2006.

[4] Yamamoto, O: Fast computation of three-dimensional convex hulls using graphics hardware. *Japan Journal of Industrial and Applied Mathematics*, vol. 22, No, 2, pp. 291-310, 2005.

[5] 河野勇人, 山本修身: 比較的少数の母点が凸領域内に作るボロノイ点の効率的数え上げについて. 平成 24 年度電気関係学会東海支部連合大会予稿集. B4-7, 2012.