Vol.2010-AL-131 No.1
2010/9/22

単純多角形に対する包含多角形列の構成法

大西　建輔,[†1]　星　守[†2]

我々は, 凸多角形に対する包含多角形列の計算手法について研究を行ってきた. その中で, 2 つの構成手法, 最適法と貪欲法を提案した. 前者は, 最小面積の k 多角形を順次計算し, 後者は最小面積の k 多角形に近い k 多角形を最適法よりも少ない領域で計算する手法である.

本稿では, より一般的な単純多角形に対する包含多角形列の計算法を提案する. これまでの実験で貪欲法は省メモリかつ最適法とほぼ同等の性能を示したため, 貪欲法に基づく方法を提案する.

Computing a Sequence of Circumscribing Polygons for a Given Simple Polygon

Kensuke Onishi, [†1] and Mamoru Hoshi [†2]

We have studied computing methods for a sequence of circumscribing polygons for convex polygon. In our previous paper, we proposed two constructive methods: optimal and greedy. The former computes the smallest k-gons, the latter quickly computes nearly smallest k-gons with small memory except for $k = 3, 4$.

In this paper, we research computing method for a sequence of circumscribing polygons for a given simple polygon, which is more general polygon rather than convex. Since the previous paper showed the greedy method needs small memory and compute circumscribing polygon nearly which is almost the same as that of the optimal method, we propose a method based on the greedy method.

†1 東海大学理学部情報数理学科, 〒 259-1292 神奈川県平塚市北金目 4-1-1, Email: onishi@tokai-u.jp
Department of Mathematical Sciences, School of Science, Tokai University, 4-1-1, Kitakaname. Hiratsuka, Kanagawa, Japan, 259-1292
†2 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications.

1. 始　め　に

我々は, 形状 (単純多角形) による画像検索を扱うためのアルゴリズム, 類似度などを考察している. 形状による画像検索では, " 検索にどのような距離, またはどのような類似度を使うのか?" という問題があり, これまで, 多くの多角形間の類似度が提案されてきた[4]. 例えば, 類似度としてよく使われる距離として, ハウスドルフ距離がある. しかし, その計算は, 必ずしも簡単でないと指摘されている[4] [p.185].

基本的な計算である類似度計算が簡単でないと, 検索サーバの計算パワーが類似度計算に割かれてしまい, サーバでの検索が著しく低下する. つまり, 類似度は簡単に計算できることが望まれる. 多角形間の類似度計算が簡単でないことの一因に, 多角形の頂点数の違いを挙げることができる. そこで我々は次の問題を考えた.

問題 n 頂点の単純多角形 P が与えられている. k 多角形の列 $P_k (k = 3, \ldots, n-1)$ を計算しなさい. ただし, それぞれの P_k は P を含むとする.

本論文では, 単純多角形に対する包含多角形列 $\{P_k\}$ を計算するアルゴリズムを提案し, その計算量の解析をおこなう (3 節).

2. 関 連 研 究

我々は[6] において, 凸多角形の包含多角形列を計算する 2 つの手法：最適法と貪欲法を提案した. n 頂点の凸多角形に対し, 最適法と貪欲法は, それぞれ $O(n^2 \log n)$ 時間と $O(n^2)$ 領域, $O(n^2)$ 時間と $O(n)$ 領域で計算を終了する. これらの手法を実際に実装し, 計算機実験を行った結果, 包含三角形と包含四角形の場合を除き, 貪欲法は, 最適法とほぼ同じ面積の包含 k 多角形を出力した.

さらに, 貪欲法を改良した**改良貪欲法**も提案した (アルゴリズム 1). この手法は, 貪欲法においてボトルネックとなっていた出力部分の大きさを制限する方法であり, 性能は貪欲法と同じである. 改良貪欲法の計算量は, $O(n \log n)$ 時間, $O(n)$ 領域である.

3. 提 案 手 法

本節では, 与えられた単純多角形 P の包含多角形列を計算する手法について説明する. 改良貪欲法を単純多角形に適用できるようにした手法をアルゴリズム 2 に示す.

アルゴリズム 2 のステップ (1) では多角形 P の頂点集合に対する凸包を計算し, その頂点数を c とする. $k = 3, \ldots, c-1$ のときは, 改良貪欲法 (アルゴリズム 1) を用い, $k = c, \ldots, n$

Vol.2010-AL-131 No.1
2010/9/22

アルゴリズム 1 凸多角形の包含多角形列を計算する改良貪欲法

Input $P = (p_0 p_1 \dots p_{n-1})$: n 頂点の凸多角形;

Output $\{P_k\}$: 多角形列, P_k は k 個の頂点を持ち, $P_k \supset P$ $(k = 3, 4, \dots, n-1)$;

(1) **for** $i := 0$ **to** $n-1$ **do**

(a) 直線 l_{i-2}, l_{i+1} の交点 q_i が頂点 p_i 側に存在するとき, $\triangle p_{i-1}p_ip_{i+1}$ の面積 S_i を計算. ただし, l_j は p_j, p_{j+1} を通る直線;

(b) S_i をキーとし, ペア (i, S_i) をヒープ H に追加;

(2) **for** $k = n-1$ **downto** c **do**

(a) H から (i, S_i) を取り出す. ただし, H の中で S_i は最小値;

(b) 添字 i と点 q_i を出力;

(c) H の $(i-1, S_{i-1})$ と $(i+1, S_{i+1})$ を更新;

のときは, アルゴリズム 2 を用い, 包含 k 多角形を計算する.

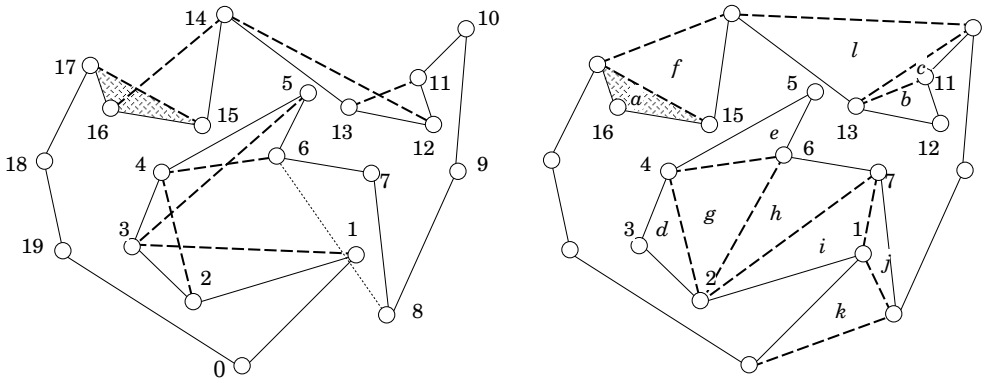

図 1 単純多角形及び外部三角形 (左) とアルゴリズム 2 の出力 (右)

図 1 を用いて, アルゴリズム 2 の動作を説明する. 図 1(左) の多角形 $P = (p_0, \dots, p_{19})$ が入力されたとする. アルゴリズム 2 のステップ (1) では, 凸包を計算し, $c = 8$ という値を得る. 次に, 凹角 $\angle p_{i-1}p_ip_{i+1}$(添字 i で表される) に対し, 線分 $\overline{p_{i-1}p_{i+1}}$ を考え, $\triangle p_{i-1}p_ip_{i+1}$ の面積 S_i を計算し, (i, S_i) をヒープ H に, S_i を $V[i]$ に保存する. 例えば, 凹角 $\angle p_1p_2p_3$(添字 2 で表される) の場合は, 線分 $\overline{p_1p_3}$ を考え, S_2 を計算し, $(2, S_2)$ を H に, S_2 を $V[2]$ に保存する.

アルゴリズム 2 単純多角形の包含多角形列を計算する貪欲法

入力 $P = (p_0 p_1 \dots p_{n-1})$: 単純多角形;

出力 $\{P_k\}$: 多角形列, P_k は k 個の頂点を持ち, $P_k \supset P$ $(k = 3, 4, \dots, n-1)$;

(1) P の凸包を計算し, c を凸包の頂点数とする;

(2) **for** $i := 0$ **to** $n-1$ **do**

(a) $V[i] := \infty$;

(b) $\angle p_{i-1}p_ip_{i+1}$ が凹であり, 線分 $\overline{p_{i-1}p_{i+1}}$ が P のどの辺と交わらないとき, $\triangle p_{i-1}p_ip_{i+1}$ の面積 S_i を計算.

(c) キーを S_i とし, ペア (i, S_i) をヒープ H に追加. また, $V[i] := S_i$;

(3) **for** $k := n-1$ **downto** c **do**

(a) **repeat**

H からペア (i, S_i) を取り出す.

until (S_i と $V[i]$ が等しい);

(b) S_i の添え字 i を出力;

(c) S_{i-1} と S_{i+1} を計算し, $(i-1, S_{i-1})$ と $(i+1, S_{i+1})$ を H に追加. また, $V[i-1] := S_{i-1}, V[i+1] := S_{i+1}$;

(4) アルゴリズム 1 を用い, P_k $(k = 3, \dots, c)$ を計算;

計算を順次進めるが, $i = 7$ のとき, 線分 $\overline{p_6p_8}$ は, P の辺 $\overline{p_0p_1}, \overline{p_1p_2}$ と交点を持つため, 面積計算をおこなわない. これは, 多角形列のどの多角形も単純多角形であることを保証するためである.

ステップ (2) が終了したとき, H に含まれる頂点は, $p_2, p_3, p_4, p_5, p_{12}, p_{13}, p_{15}, p_{16}$ となる. H での最小面積 S_{16} を含むノード $(16, S_{16})$ をヒープから取り出し, 添字 16 を出力する. S_{16} の面積を持つ三角形は, 図 1(右) では"a" で表されている. 次に, $\triangle p_{14}p_{15}p_{17}$(添字 15 で表される) の面積 S'_{15} を計算し, H に $(15, S'_{15})$ を追加し, S'_{15} を $V[15]$ に保存する. また, $\angle p_{15}p_{17}p_{18}$(添字 17 で表される) は, 凸角なので, 面積は計算せずに, $V[17]$ に ∞ を代入し, 終了する.

これを繰り返すことで, 出力として, $16, 12, 11, 3, 5, 15, 4, 6, 2, 7, 1, 13$ という添字の列を得

ることができる. この出力に対する単純多角形列を図示したものが図 1(右) である. つまり, 包含多角形列の中で, 頂点数 19 の多角形 P_{19} は, 元の単純多角形から p_{16} を除いたものである. 多角形 P_{19} は, 単純多角形であり, 元の多角形を包含する. 次の 18 頂点の多角形 P_{18} は, 多角形 P_{19} から, p_{12} を除いたものである.

別の見方をすると, アルゴリズム 2 は単純多角形の外部に三角形を加えていき, 凸包にする手続きとも言える. この例では, 20 頂点の多角形に 12 個の三角形をアルファベットの順に加えていき, 8 頂点の凸包を得ることになる. 凸包に到達したあとは, アルゴリズム 1 を用い, 凸多角形に対する包含多角形を計算する.

次にアルゴリズム 2 の計算量を解析する. $V[i]$ は計算途中の三角形 $\triangle p_{i-1}p_ip_{i+1}$ の面積を保持する配列である. 凸多角形の包含多角形列を計算する場合には, ヒープに含まれる三角形の面積 S_{i-1}, S_{i+1} が常に増大していくため, 三角形の面積をヒープだけで管理することが可能であった[6]. しかし, 単純多角形の包含多角形を計算する場合には, 三角形の面積が大きくなる場合と, 小さくなる場合がある (図 2). 図 2 では, 左側の点線は線分 $\overline{p_{i+1}p_{i+2}}$ に平行な

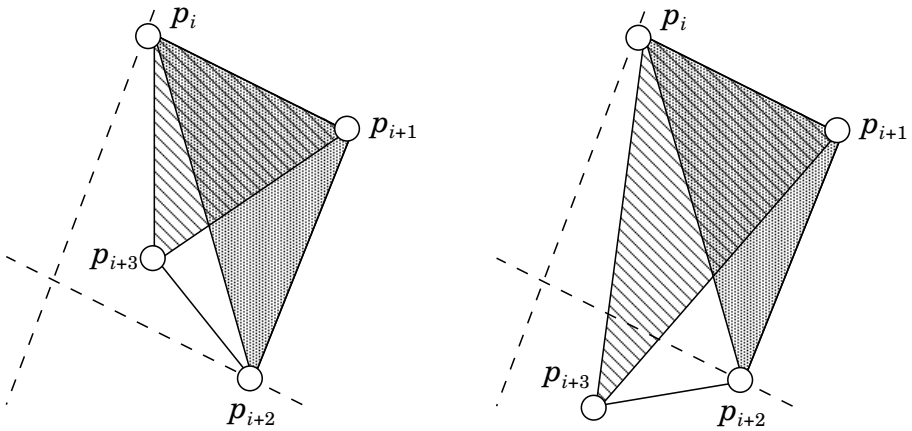

図 2 三角形の面積が減少する場合 (左) と増加する場合 (右)

直線であり, 下側の点線は線分 $\overline{p_ip_{i+1}}$ に平行な直線である. このとき, 三角形 $\triangle p_ip_jp_k$ の面積を $S_{i,j,k}$ で表すと, どちらの場合でも, $S_{i+1,i+2,i+3} < S_{i,i+1,i+2}$ が成り立つ. ここで, アルゴリズム 2 の出力として $i+2$ が出力されると, $\triangle p_ip_{i+1}p_{i+3}$ の面積を計算し, $\triangle p_ip_{i+1}p_{i+2}$ の面積を更新する必要がある. 図 2(左) の場合は $S_{i,i+1,i+2} > S_{i,i+1,i+3}$ が, 図 2(右) の場合は $S_{i,i+1,i+2} < S_{i,i+1,i+3}$ が成り立つ.

そのため, ヒープだけで三角形の面積を管理することができずに, 現在の面積の値を保持する配列 $V[\,]$ を用い, 添字が i のとき, $V[i]$ と一致する値だけを使う (アルゴリズム 2 のステップ (3)(a)). このとき, ヒープには必ず現在の値 S_i が含まれているため, **repeat** 文は必ず終了する. ステップ (3)(c) でヒープに要素を追加しているが, 増加する要素は高々$2n$ 個であるため, オーダーとしてみると計算時間は変化しない. よって, ヒープを使い, データを管理する部分は, 作成に $O(n\log n)$ 時間と $O(n)$ 領域が必要となる. また, データの追加は $O(\log n)$ 時間で済み, それを高々$2n$ 回繰り返すため, $O(n\log n)$ 時間が必要となる.

次に, アルゴリズム 2 のステップ (2)(a) とステップ (3)(c) で行われるの交点の計算回数の評価を行う. 直接的に計算を行うと, ステップ (2)(a) での計算回数は, $O(n^2)$ となる. 次の補題を用いることで, 計算回数を減少させることができる.

補題 1 P を単純多角形とする. P の凸包 $CH(P)$ を考える. $CH(P)$ の辺は, P の辺を含むか, P の辺と端点で交わる.

証明: 凸包の辺 e を含む直線 l を考える. 凸包の性質から, P の頂点は l 上か, l の片側に存在する. l の片側にある 2 点を結ぶ P の辺は, e と交わらない. また, l 上の点と l 上にない点を結んだ P の辺は, l と端点で交わる. さらに, l 上の 2 点を結んだ P の辺は, e そのものになるか, e に含まれる P の辺となる. □

アルゴリズム 2 では, 最初に凸包を計算しているため, どの点が凸包の点であるかを知ることができる. そのため, 凸包上の点であるかどうかを記憶する配列を持つことで, その情報を利用可能である.

さらに次の補題が成り立つ.

補題 2 単純多角形 $P = (p_0, \ldots, p_{n-1})$ とその凸包 $CH(P)$ を考える. $p_i, p_j (i < j)$ を $CH(P)$ 上の隣接する 2 頂点とする.
このとき, 鎖 $p_i, p_{i+1}, \ldots, p_j$ と辺 $\overline{p_ip_j}$ は, 単純多角形 $P' = (p_i, p_{i+1}, \ldots, p_j, p_i)$ をなす. また, $p_k, p_l \in P'$ に対し, $\overline{p_kp_l}$ が P' のどの辺とも交わらないならば, $\overline{p_kp_l}$ は P のどの辺とも交わらない.

証明: P が単純多角形であるため, $p_i, p_{i+1}, \ldots, p_j$ に含まれる線分は, 端点以外で交わりを持たない. また, $\overline{p_ip_j}$ は凸包の辺であるため, 補題 1 から, P の辺とは端点を除き交わらないか, P の辺を含む.

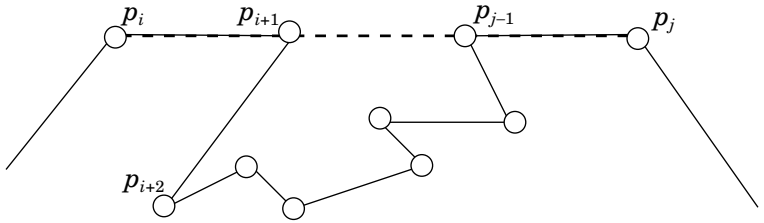

図 3 凸包の辺 $\overline{p_ip_j}$ が P の 2 つの辺を含む場合

P の辺と交わらない場合, その部分集合である P' の辺とも交わらない. よって, P' も単純多角形となる. P の辺を含む場合は, さらに 2 つの場合に分けられる. 線分 $\overline{p_ip_j}$ が P でも隣接する場合と, そうでない場合である. 隣接する場合は, p_i と p_j の間に点が存在しない場合であり, 単なる辺となる,

そうでない場合は, $\overline{p_ip_j}$ 上に 2 つ以上の P の辺が含まれる場合である (図 3 参照). この場合は, 辺上に含まれる P の点を除くと, 鎖の中に 1 辺を加えると単純多角形となる部分が存在する. 図 3 の場合は, $\overline{p_ip_j}$ が凸包の辺とすると, $P' = (p_{i+1}, p_{i+2}, \ldots, p_{j-1}, p_{i+1})$ とすれば, P' が単純多角形となる.

2 つ目のステートメントの対偶を証明する. つまり, 次を示す.

$\overline{p_kp_l}$ が P のある辺と交わるならば, $\overline{p_kp_l}$ が P' のある辺と交わる.

仮定で示される辺が P' の辺ならば, 結論が示される. そうでない場合を考える. 仮定からある辺 $\overline{p_mp_{m+1}}$ $(m \notin [i, j])$ が存在し, $\overline{p_kp_l}$ と交わる. この交点が, P' の内側に存在するとすると, p_m, p_{m+1} が P' の頂点となり, 矛盾となる. よって, この交点は, P' の外側にあることになる. このとき, $\overline{p_kp_l}$ は P' の内部から外部にでる必要があるため, P' のある辺と必ず交わる. □

補題 2 から, それぞれの部分多角形 P' を独立に扱うことができる. 図 1 の場合は, $(p_0, p_1, \ldots, p_8)$, $(p_{10}, p_{11}, \ldots, p_{14})$, $(p_{14}, p_{15}, p_{16}, p_{17})$ という 3 つの部分多角形を考えればよいことになる.

よって, アルゴリズム 2 の計算量を示すことができた.

定理 1 アルゴリズム 2 を用いることで, n 頂点の単純多角形の包含多角形列は, $O(n_s^2)$ 時間と $O(n)$ 領域で計算ができる. ただし, n_s は補題 2 の P' の頂点数の最大とする.

4. ま と め

本稿では, 単純多角形を含む多角形列を計算する手法とその計算時間の解析について述べた. 今回の提案手法は, $O(n_s^2)$ 時間と $O(n)$ 領域で終了する. しかし, n_s は最悪の場合, n となるため, その計算時間は, $O(n^2)$ と変わらない.

線分間の交点の計算回数は, 高々$O(n \log n)$ と予想され, 現在その解析を行っている最中である.

また, 今回の提案手法をもとに, 単純多角形の間の類似度を提案することも考えている.

謝　　辞

本研究の一部は, 日本学術振興会科学研究費基盤研究 (C) No.22500097 の助成を受けた.

参 考 文 献

1) A.Aggrawal and J.K.Pach. Note on searching in multidimensional monotone arrays. In *Proc. 29th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 479-512, 1988.
2) L.Boxer, C.-S.Chang, R.Miller and A. Rau-Chaplin. Polygonal approximation by boundary reduction, *Pattern Rec. Letters* **14**, pages 111-119, 1993.
3) P.Brass. On the approximation of polygons by subpolygons. In *Proc. European Workshop Comput. Geom. (EuroCG)*, pages 59-61, 2000.
4) J.E.Goodman and J.O'Rourke ed. Handbook of Discrete and Computational Geometry, second edition, Chapman & Hall, 2004.
5) P.M.Gruber. Approximation of Convex Bodies. In *Handbook of Convex Geometry*, Vol. A (P.M.Gruber and J.M.Wills, ed.), North-Holland, pages 319-345, 1993.
6) K.Onishi and M.Hoshi. Computing a Sequence of Circumscribing Polygons for Convex Polygon, In *Proc. of Computational Geometry and Discrete Mathematics,* **数理解析学研究所講究録** *1641*, pages 90-98, 2009.