Contact Zoneを用いたボロノイ図作成の並列化による処理時間短縮手法

岡鼻　雄飛[1]　後藤　佑介[1]

概要：近年，Global Positioning System (GPS) を用いて目的地検索やカーナビゲーションを行う位置情報サービスが注目されている．位置情報サービスでは，移動端末による空間軸上の変化を表現できる多次元空間上で，クエリとなるオブジェクトが自身からもっとも近いオブジェクトを探索する最近傍探索が用いられるが，クエリからすべてのオブジェクトまでの距離をそれぞれ算出した上で探索する必要があり，探索するオブジェクト数の増加にともない計算量は膨大となる．この計算量を削減するため，ボロノイ図を用いた領域分割による探索手法が提案されている．ボロノイ図を用いた領域分割手法では，オブジェクト間の位置関係をもとに各オブジェクトに対してボロノイ領域を作成することで，最近傍探索における計算量を削減できる．また，Contact Zone を用いてボロノイ図の作成にかかる処理時間を短縮する手法があるが，オブジェクトに対するボロノイ領域の作成は一つずつ順番に行われるため，オブジェクト数の増加にともない処理時間は長大化する．そこで，本研究では，Contact Zone を用いたボロノイ図作成の並列化による処理時間の短縮手法を提案する．提案手法では，Contact Zone を用いて複数のオブジェクトに対するボロノイ領域の作成処理を並列化することで，ボロノイ図の作成にかかる時間を短縮する．評価の結果，提案手法は，並列化を行わない既存のボロノイ図作成手法と比較して，ボロノイ図の作成にかかる時間を約 15.9%短縮できることを確認した．

1. はじめに

近年，Global Positioning System (GPS) を用いて目的地検索やカーナビゲーションを行う位置情報サービスが注目されている．位置情報サービスでは，移動端末による空間軸上の変化を表現できる多次元空間上で，クエリオブジェクト (以下，クエリ) が自身からもっとも近いオブジェクトを探索する最近傍探索を用いることが一般的である．最近傍探索を行う場合，クエリからすべてのオブジェクトまでの距離をそれぞれ算出した上で探索する必要があり，オブジェクトの増加にともない探索に必要となる計算量は膨大となる．この計算量を削減するため，ボロノイ図を用いた領域分割手法 [1], [2], [3]，およびデータ構造を用いた空間アクセス手法 [4], [5], [6] が提案されている．

ボロノイ図を用いた領域分割手法では，空間上に複数のオブジェクトが存在する場合，各オブジェクトに対して自身がもっとも近いオブジェクトとなる領域 (以下，ボロノイ領域) を作成し，空間を分割する．このとき，オブジェクト間の位置関係をもとに，各オブジェクトに対してボロノイ領域を作成することで，最近傍探索における計算量を削減できる．また，ボロノイ図の作成にかかる時間をさらに短縮する方法として，空間上のある点を中心としてオブジェクトまでの距離を半径とする円の領域である Contact Zone を用いる手法 [7], [8] が挙げられる．この手法では，Contact Zone を用いて空間上に存在するオブジェクトに対するボロノイ領域を作成するが，オブジェクトを一つずつ順番に選択するため，オブジェクト数の増加にともない処理時間は長大化する．

本研究では，Contact Zone を用いたボロノイ図の作成において，各オブジェクトで行うボロノイ領域の計算を並列化することでボロノイ図の作成時間を短縮する手法を提案する．提案手法では，Contact Zone を用いて複数のオブジェクトに対するボロノイ領域の作成処理を並列化することで，並列化を行わない既存手法と比較してボロノイ図の作成にかかる時間を短縮する．

2. ボロノイ図

2.1 概要

空間上に複数のオブジェクトが存在する場合，各オブジェクトに対してボロノイ領域を作成し，空間を分割することでボロノイ図を作成する．ボロノイ図を作成すること

[1] 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

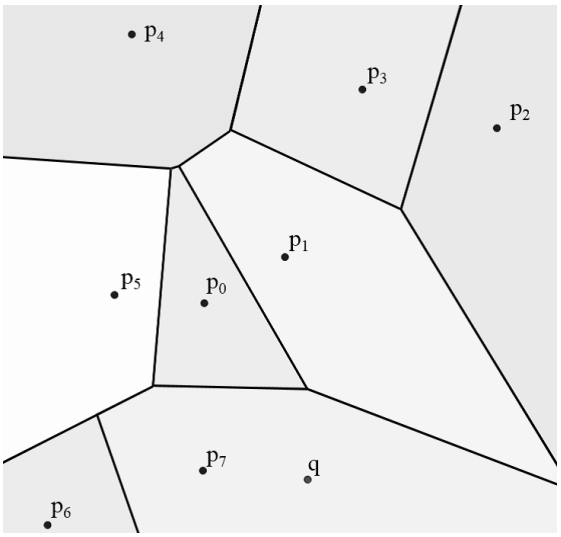

図 1 空間におけるボロノイ図の作成例

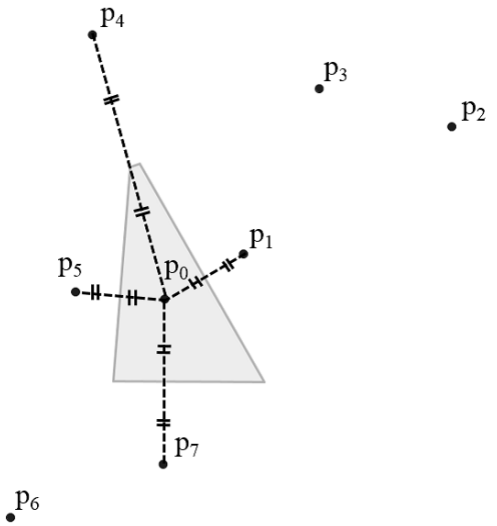

図 2 ボロノイ領域の作成例 (単純手法)

で，クエリは自身の最近傍となるオブジェクトの探索にかかる処理時間を短縮できる．ここで，ボロノイ領域を構成する辺をボロノイ辺，および頂点をボロノイ頂点と呼ぶ．

ボロノイ図の作成例を**図 1** に示す．オブジェクト $p_0, ..., p_7$ が空間上に存在するとき，任意の二つのオブジェクト間における垂直二等分線をもとに，空間を各オブジェクトに対応する複数のボロノイ領域に分割することでボロノイ図を作成する．作成したボロノイ図をもとにクエリ q に対する最近傍探索を行う場合，初めにオブジェクトをランダムに一つ選択する．p_0 を選択した場合，p_0 に隣接するボロノイ領域内のオブジェクトのうち，クエリ q との距離がもっとも近いオブジェクトである p_7 を選択する．このとき，クエリ q は p_7 のボロノイ領域の内部であるため，クエリ q からもっとも近いオブジェクトは p_7 となり，探索を終了する．

2.2 ボロノイ図の作成手法

単純なボロノイ図の作成手法 [1](以下，単純手法) では，選択したオブジェクト (以下，ターゲット) とターゲット以外のオブジェクトとの間で垂直二等分線を引くことで，ターゲットに対するボロノイ領域を作成する．空間上にオブジェクト $p_0, ..., p_7$ が存在する場合，選択したターゲット p_0 に対するボロノイ領域の作成例を**図 2** に示す．p_0 と p_0 以外のオブジェクト $p_1, ..., p_7$ との間に垂直二等分線をそれぞれ引き，作成した p_0 を含む半平面の共通部分で構成される多角形の領域をボロノイ領域とする．図 2 の場合，p_0 のボロノイ領域は，p_0 に対する p_1, p_4, p_5, p_7 との 4 本の垂直二等分線で囲まれた四角形の領域となる．

単純手法では，ボロノイ図の作成にかかる計算量が空間上に存在するオブジェクト数の 3 乗に比例するため，オブジェクト数が増加すると計算量は膨大になる．このため，計算量を削減するボロノイ図の作成手法は必要である．

3. 関連研究

3.1 概要

これまでに，空間上に存在するオブジェクトに対してボロノイ図を作成する手法はいくつか提案されている．本章では，既存のボロノイ図作成手法として，2*farthest vertex 法 [7]，逐次添加法 [9]，および分割統治法 [10] を順番に説明する．

3.2 2*farthest vertex 法

2*farthest vertex 法 [7] では，初めに，空間上に存在するオブジェクトに対して，ターゲットからの距離が近いオブジェクトから順番にターゲットとの垂直二等分線を引く．初期領域を作成できる場合，初期領域を構成する頂点のうちターゲットからの距離がもっとも遠い頂点を一つ選択する．次に，中心がターゲットの位置，半径がターゲットと選択した頂点間の距離の 2 倍となる円を作成する．このとき，ターゲットは作成した円の内側に存在するオブジェクトに対してターゲットとの垂直二等分線を引き，ボロノイ領域を更新する．一方で，初期領域を作成できない場合，単純手法と同様の手順でボロノイ領域を作成する．

2*farthest vertax 法を用いてボロノイ領域を作成する例を**図 3** に示す．ターゲットは p_0 とし，オブジェクト $p_1, ..., p_7$ は空間上に存在する．まず，ターゲット p_0 からの距離が近い p_5, p_1, p_7 の順番に p_0 との垂直二等分線を引き，p_0 の初期領域となる三角形を作成する．次に，中心が p_0，半径が p_0 と p_0 からもっとも遠い頂点 v との距離の 2 倍となる円を作成する．p_0 は，作成した円の内側に存在するオブジェクトのうち，p_0 との距離がもっとも近い p_3 と

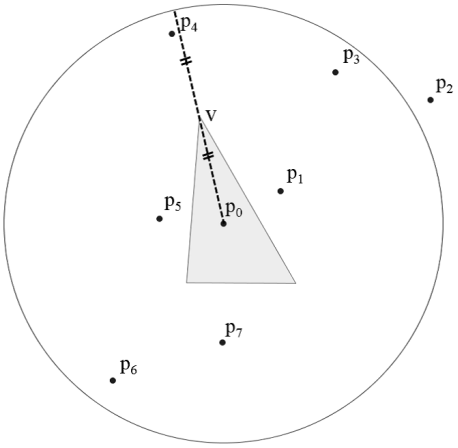

図 3 ボロノイ領域の作成例 (2*farthest vertex 法)

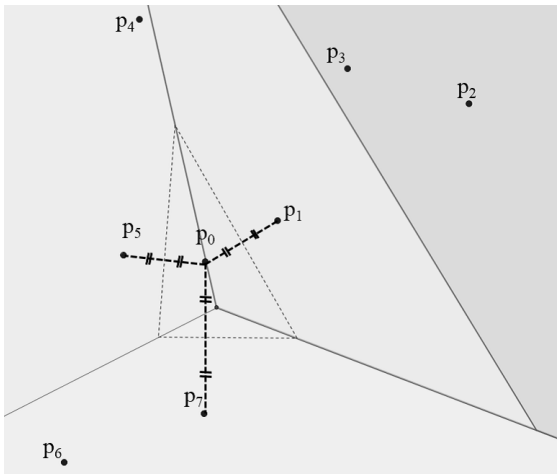

図 4 ボロノイ領域の作成例 (逐次添加法)

の間で垂直二等分線を引き，ボロノイ領域を更新する．一方で，p_2 は作成した円の外側に存在するため，ボロノイ領域の作成に影響せず除外でき，単純手法に比べて計算量を削減できる．

3.3 逐次添加法

逐次添加法 [9] では，空間上に存在する複数のオブジェクトに対して，初めに一定数のオブジェクトを選択する．次に，選択したオブジェクトに対して，単純手法を用いてボロノイ図を作成する．この後，初めに選択しなかった残りのオブジェクトを用いてボロノイ領域を順番に更新し，すべてのオブジェクトに対するボロノイ図を作成する．

逐次添加法でボロノイ領域を作成する例を**図 4** に示す．空間上に存在するオブジェクト $p_0, ..., p_7$ のうち 4 個のオブジェクト p_1, p_2, p_5, p_7 を選択する場合を考える．まず，選択した 4 個のオブジェクトに対するボロノイ図を作成する．次に，初めに選択しなかった残り 4 個のオブジェクト p_0, p_3, p_4, p_6 に対するボロノイ領域を順番に更新し，すべてのオブジェクトに対するボロノイ図を作成する．ここで，p_0 を選択する場合，p_0 からもっとも近いオブジェクトである p_5 を求め，p_0 と p_5 との間に垂直二等分線を引く．これまでに選択したオブジェクトに対するボロノイ領域を構成するボロノイ辺のうち，この垂直二等分線と交差する辺をボロノイ領域の境界とするオブジェクトを求め，このオブジェクトとターゲットとの間で垂直二等分線を引き，ボロノイ領域を更新する．図 6 の場合，p_0 と p_5 との間に引いた垂直二等分線は，p_1 および p_7 のボロノイ領域を構成する辺とそれぞれ交差するため，p_0 と p_1，および p_0 と p_7 との間でそれぞれ垂直二等分線を引き，p_0 のボロノイ領域を作成する．以上の手順で，残りのオブジェクト p_3, p_4, p_6 に対するボロノイ領域を作成し，ボロノイ図の作成を終了する．このとき，p_2 を構成するボロノイ辺は，p_0 と p_1, p_5, p_7 との間でそれぞれ引いた三本の垂直二等分線と交差しないため，ボロノイ領域の更新に影響せず，単純手法に比べて計算量を削減できる．逐次添加法における計算量は，最悪の場合 $O(n^2)$ である．

3.4 分割統治法

分割統治法 [10] では，初めに空間上で x 軸に垂直な直線を任意の数だけ引いて分割し，分割した領域ごとに逐次添加法を適用してボロノイ図を作成する．次に，各領域で作成した複数のボロノイ図を順番に組み合わせて更新することで，空間全体に対するボロノイ図を作成する．

分割統治法でボロノイ図を作成する例を**図 5** に示す．オブジェクト $p_1, ..., p_7$ は空間上に存在し，太線で示した x 軸に垂直な直線を一本引き空間を二分割する場合を考える．分割統治法では，二分割した領域ごとに逐次添加法を適用してボロノイ図を作成する．次に，各領域で作成した二つのボロノイ図を組み合わせて更新することで，図 1 に示したボロノイ図となる．

分割統治法の計算量は，最悪，および平均のどちらにおいても $O(nlogn)$ である．領域ごとにボロノイ図を計算する処理を並列化し，処理速度を向上させる手法も存在するが，複数のボロノイ図を組み合わせて更新するときに複雑な処理が必要となる．

4. Contact Zone を用いたボロノイ図の作成手法

4.1 概要

ボロノイ図の作成にかかる計算量を削減する手法として，Contact Zone (以下，CZ) を用いたボロノイ図の作成手法 [7][8] を説明する．CZ は，初めに設定した点を中心として，この点からターゲットまでの距離を半径とする円である．円の中心となる点の設定方法は，ターゲットとオブジェクトによる複数の垂直二等分線で構成される多角形の

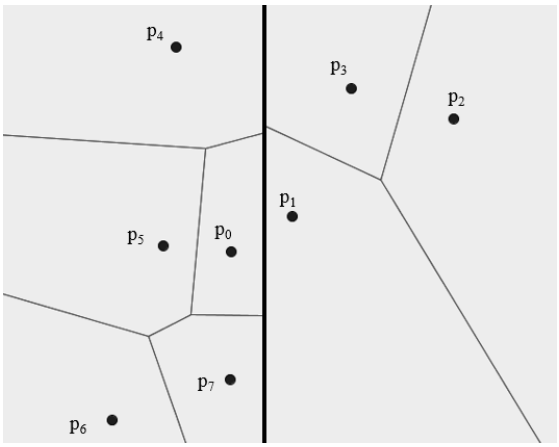

図 5 ボロノイ領域の作成例 (分割統治法)

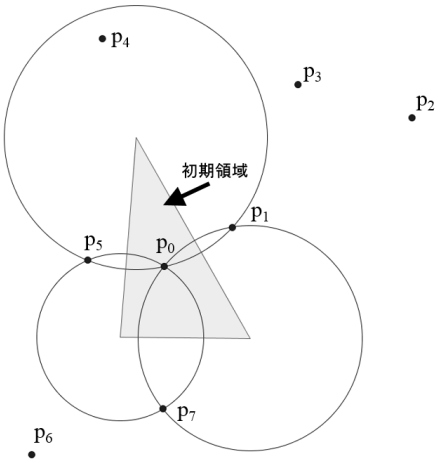

図 6 ボロノイ領域の作成例 (初期領域を考慮して CZ を用いる手法)

領域 (以下，初期領域) を考慮するか否かで異なる．以下で，初期領域を考慮する場合と考慮しない場合のそれぞれについて，CZ を用いたボロノイ図の作成手順を説明する．CZ を用いる手法における計算量は，最悪の場合 $O(n^3)$ である．

4.2 初期領域を考慮する場合

まず，単純手法と同様に，ターゲットから空間上に存在する複数のオブジェクトに対して，ターゲットからの距離が近いオブジェクトから順番にターゲットとの垂直二等分線を引く．複数の垂直二等分線で初期領域を作成できる場合，中心が初期領域を構成する多角形の頂点，半径が頂点からターゲットまでの距離となる CZ の円を頂点ごとに複数作成し，これらの CZ の内側に存在する各オブジェクトに対して，ターゲットとの垂直二等分線を引く．このとき，CZ の内側に存在しないオブジェクトはボロノイ領域の作成に影響しないため，ボロノイ図の作成における計算から除外できる．一方で，初期領域を作成できない場合，単純手法と同様の手順でボロノイ領域を作成する．

初期領域を考慮した上で CZ を用いてボロノイ領域を作成する場合の例を**図 6** に示す．ターゲットを p_0 とし，オブジェクト $p_1, ..., p_7$ が空間上に存在する．まず，p_0 からの距離が近い順番に，p_0 との垂直二等分線を引く．このとき，p_0 から p_5, p_1, p_7 に対してそれぞれ引いた三本の垂直二等分線で構成される三角形を初期領域とする．次に，初期領域を構成する三角形の各頂点を中心とし，各頂点から p_0 までの距離が半径となる CZ の円を三つ作成する．このとき，オブジェクト p_2, p_3, p_6 は三つの CZ の外側に存在するため，ボロノイ領域の作成における計算から除外でき，単純手法に比べて計算量を削減できる．以上より，ターゲット p_0 に対するボロノイ領域を作成できる．これらの手順をすべてのオブジェクトに適用することで，ボロノイ図を作成する．

4.3 初期領域を考慮しない場合

次に，初めから初期領域を考慮せずに，CZ を用いてボロノイ図を作成する手法を説明する．まず，ターゲットとの距離が近いオブジェクトから順番に，ターゲットとの垂直二等分線を引く．ターゲットと各オブジェクトとの間で作成する複数の垂直二等分線のうち二本で交点 C を作成できる場合，ターゲットからこれら二本の垂直二等分線にそれぞれ引いた垂直な直線で分けられた二つの領域のうち，交点 C を含む側の領域を Pruning Region (以下，PR) と設定する．次に，交点 C を中心とし，ターゲットから交点 C までの距離を半径とする円 CZ を作成する．このとき，PR の外側，および「PR の内側かつ CZ の内側」のどちらかの条件を満たす領域に存在するオブジェクトに対して，ターゲットとの間で垂直二等分線を引き，ボロノイ領域を作成する．一方，「PR の内側かつ CZ の外側」を満たす領域に存在するオブジェクトは，ボロノイ領域の作成に影響しないため除外する．

初期領域を考慮せずに PR と CZ を用いてボロノイ領域を作成する場合の例を**図 7** に示す．ターゲットを p_0 とし，オブジェクト $p_1, ..., p_7$ が空間上に存在する．まず，p_0 から距離が近いオブジェクトから順番に p_1, p_5 を選択し，p_0 との垂直二等分線をそれぞれ引く．これら二本の垂直二等分線の交点を C として，p_0 と p_1，および p_0 と p_5 を通るそれぞれ二本の直線で分けられる領域のうち，交点 C を含む側の領域を PR とする．次に，中心を C，半径を C から p_1 もしくは p_5 までの距離とする CZ の円を作成する．このとき，オブジェクト p_4 は「PR の内側かつ CZ の内側」に存在するため，p_0 と p_4 との間に垂直二等分線を引き，ボロノイ領域を更新する．また，PR の外側に存在するオブジェクトのうち，ターゲットからの距離がもっとも短いオブジェクト p_7 に対して，ターゲットとの間で垂直二等分線を引き，ボロノイ領域を更新する．なお，オブジェク

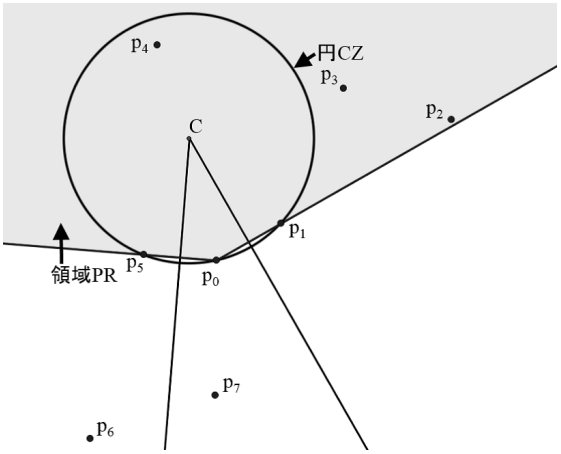

図 7 ボロノイ領域の作成例 (初期領域を考慮せずに CZ を用いる手法)

ト p_2, p_3 は「PR の内側かつ CZ の外側」に存在するため，ボロノイ領域の作成に影響せず除外でき，単純手法と比べて計算量を削減できる．

5. 並列計算の導入

2 章で説明したように，既存のボロノイ図作成手法では，空間上に存在するオブジェクト数が増加すると，ボロノイ図の作成にかかる処理時間が長大化する問題があった．本研究では，この処理時間を短縮するため，複数のターゲッ値に対するボロノイ領域の作成処理を並列化する手法を考える．

まず，既存のボロノイ図作成手法において，ボロノイ領域の作成処理の並列化が可能か検討する．単純手法，初期領域を考慮して CZ を用いる手法，初期領域を考慮せず CZ を用いる手法および 2*farthest vertex 法では，ボロノイ領域を作成するアルゴリズムがターゲットごとに独立しており，並列化は可能である．また，分割統治法では，分割した領域ごとに並列化が可能であり先行研究が存在する [10]．一方，逐次添加法では，ボロノイ図を逐次的に更新するため，並列化は不可能である．

次に，作成処理の並列化が可能な 4 種類の既存手法で発生する計算量を比較する．ボロノイ図作成の処理を並列化する場合，処理はターゲットごとに独立しているため，単純に計算量が少ない手法を選択することで処理時間を短縮できる．ボロノイ図の作成にかかる計算量を比較すると，初期領域を考慮せずに CZ を用いる手法における場合の計算量が一番少ない．以上より，初期領域を考慮せずに CZ を用いる手法を用いて，ボロノイ図作成における処理を並列化する．

6. 提案手法

6.1 概要

ターゲットごとのボロノイ領域の作成処理を並列化することで，ボロノイ図作成にかかる処理時間を短縮する手法を提案する．提案手法では，初期領域を考慮せずに CZ を用いる手法をもとに，並列でボロノイ領域を作成するターゲットの数 (以下，並列数) を設定し，各ターゲットで発生するボロノイ領域の作成処理を並列化することで処理時間を短縮する．

並列計算の実現方法として，MapReduce を用いた手法 [11] が挙げられるが，複数の計算機を用意して環境を作成する必要があるため，今回は単独の計算機で並列化が可能であるマルチスレッドを用いた並列計算を実現する．

6.2 並列処理の手順

各ターゲットに対するボロノイ領域の作成手順，および各ターゲットで作成したボロノイ領域を用いたボロノイ図の作成手順を以下に示す．

6.2.1 ボロノイ領域の作成手順

提案手法におけるボロノイ領域の作成手順は，初期領域を考慮せず CZ を用いる手法による作成手順と同じである．以下に箇条書きで示す．

(1) 未選択のオブジェクトのうち，ターゲットからの距離がもっとも近いオブジェクトを選択する．
(2) ターゲットと選択したオブジェクトとの垂直二等分線を引く．
(3) (2) で作成した垂直二等分線で交点を作成できる場合は (4) へ．作成できない場合，(1) に戻る．
(4) 作成した交点をもとに CZ と PR を作成する．
(5) PR の外側，および「PR の内側かつ CZ の内側」のどちらかに存在するオブジェクトが存在する場合，ターゲットからの距離がもっとも近いオブジェクトを選択し，(2) に戻る．一方で該当するオブジェクトが存在しない場合，処理を終了する．

6.2.2 ボロノイ図の作成手順

ボロノイ図の作成手順を以下に示す．

(1) 空間上に存在するオブジェクトのうち，あらかじめ設定した並列数と同じ数のオブジェクトをターゲットとしてランダムに選択する．
(2) (1) で選択した各ターゲットに対して，6.2.1 項で説明したボロノイ領域の作成手順に従い，並列にボロノイ領域を作成する．このとき，選択したすべてのターゲットでボロノイ領域が作成されるまで待機する．
(3) すべてのオブジェクトがターゲットとしてボロノイ領域を作成するまで (1) から (3) を繰り返す．

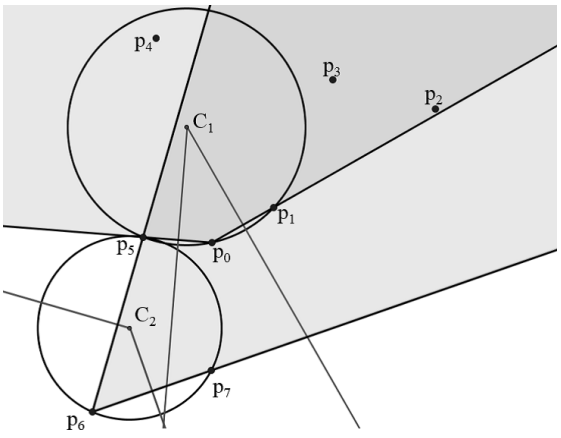

図 8 提案手法におけるボロノイ領域の作成例

6.3 提案手法の具体例

6.2 節で説明した提案手法における並列処理の手順をもとに，並列数が 2 の場合，すなわち二つのオブジェクトをターゲットとして同時に選択する場合にボロノイ領域を並列に作成する例を**図 8** に示す．空間上にオブジェクト $p_0, ..., p_7$ が存在するとき，p_0 および p_6 をターゲットとして選択し，ボロノイ領域を同時に作成する．ボロノイ領域の作成手法は，初期領域を考慮せずに CZ を用いる手法を使用する．p_0 は，p_0 からの距離が近いオブジェクトである p_5, p_1 の順番に，p_0 との垂直二等分線をそれぞれ引く．p_0 と p_5，および p_0 と p_1 との垂直二等分線の交点を C_1 とし，CZ および PR を作成する．同様に，p_6 は，p_6 からの距離が近いオブジェクトである p_7, p_5 の順番に，p_6 との垂直二等分線を引く．p_6 と p_7，および p_6 と p_5 による二本の垂直二等分線の交点を C_2 とし，CZ および PR を作成する．図 8 において，PR は灰色領域で示す．p_0 および p_6 のボロノイ領域を作成した後，未選択のオブジェクトからターゲットを二つ選択し，引き続きボロノイ領域を作成する．以上の処理手順を未選択のオブジェクトが無くなるまで繰り返すことで，ボロノイ図を作成する．

7. 評価

7.1 概要

提案手法の有用性を検証するため，計算機上でシミュレーション評価を行った．提案手法および既存手法のプログラムは，C++で実装した．なお，並列処理はスレッドを用いて実装した．評価に使用した計算機の性能を**表 1** に示す．

7.2 評価環境

提案手法の評価環境を以下に示す．

- オブジェクト間の距離は，ユークリッド距離で表す．
- 空間上に存在するオブジェクトの数は，論文 [8] を参考にして，50, 100, 150, 200, 250, 300, 350, 400, 450, 500 の 10 種類とする．
- 空間の大きさは，縦 1000，横 1000 の二次元領域とする．本研究では，平面上における探索技術への応用を想定している．
- 提案手法では，スレッドを切り替えるときのオーバヘッド，および評価で使用する計算機の同時処理可能なスレッド数 (4 スレッド) を考慮して，並列数を 5 とする．

表 1 評価に使用した計算機の性能

OS	Ubuntu 14.04 LTS
使用 CPU	Intel Core i3-2367M Processor
CPU コア数	2
スレッド数	4
メモリ	4GB
HDD	120GB

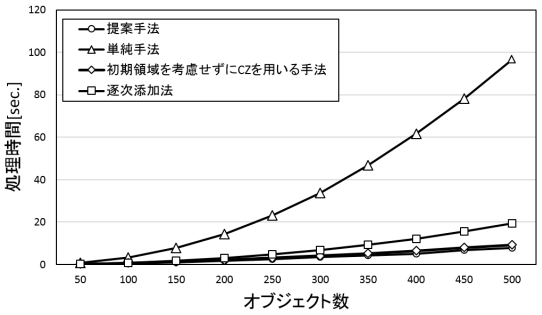

図 9 ボロノイ図の作成にかかる処理時間

7.3 オブジェクト数とボロノイ図作成の処理時間

オブジェクト数に応じてボロノイ図の作成にかかる処理時間の変化について，評価結果を**図 9** に示す．横軸は空間上に存在するオブジェクトの数，縦軸はボロノイ図の作成にかかる処理時間の平均である．評価に用いる手法は，提案手法，単純手法，初期領域を考慮せず Contact Zone を用いる手法，および逐次添加法の 4 種類とする．評価では，ボロノイ図の作成にかかる処理時間を 10 回計測し，平均値を算出した．

図 9 より，提案手法でボロノイ図の作成にかかる処理時間は，他の比較手法より短い．提案手法は，初期領域を考慮せずに CZ を用いる手法を並列化することで，処理時間を短縮する．提案手法および初期領域を考慮せずに CZ を用いる手法の二つについて，オブジェクト数が 100 の場合，提案手法の処理時間は 0.562 秒，初期領域を考慮せずに CZ を用いる手法の処理時間は 0.669 秒となり，処理時間を約 15.9%短縮する．なお，提案手法における処理時間の分散は約 0.002652 と小さい．

次に，空間上に存在するオブジェクト数をより多くした場合について，ボロノイ図の作成にかかる処理時間の評価結果を**図 10** に示す．横軸は，ボロノイ図の作成にかかる

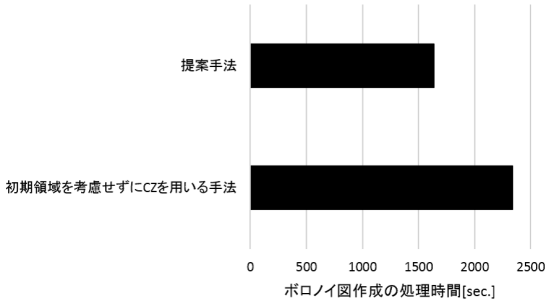

図 10 ボロノイ図作成の平均処理時間 (オブジェクト数:10000)

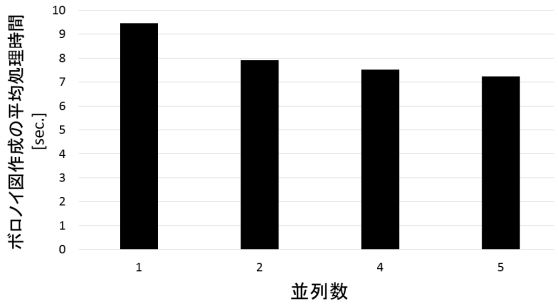

図 11 並列数とボロノイ図作成の平均処理時間 (オブジェクト数:500)

処理時間である．評価に用いる手法は，提案手法，および初期領域を考慮せずに CZ を用いる手法の二つとする．評価では，オブジェクト数が 10000 の場合におけるボロノイ図の作成にかかる処理時間を 10 回計測し，平均値を算出した．

図 10 より，空間上に存在するオブジェクト数が多くなると，オブジェクト数が 500 以下の場合と比べて，提案手法による並列化の効果が大きくなる．ボロノイ図の作成にかかる処理時間は，提案手法で 1635.2 秒，初期領域を考慮せずに CZ を用いる手法で 2338.5 秒となり，提案手法は既存手法に比べて処理時間を約 30.0%短縮できる．

7.4 並列数とボロノイ図作成の平均処理時間

並列数に応じてボロノイ図の作成にかかる処理時間の変化について，評価結果を**図 11** に示す．縦軸はボロノイ図の作成にかかる平均処理時間，横軸は並列数である．評価では，オブジェクト数が 500 の場合におけるボロノイ図の作成にかかる処理時間を 10 回計測し，平均値を算出した．

図 11 より，並列数の増加にともない平均処理時間は短くなる．今回評価に使用した計算機は 4 スレッドを同時に処理できるため，並列数が 4 の場合，処理時間は短くなる．また，並列数が 5 の場合，各スレッドの処理が同時に終了せず，6.2.2 節で説明した待ち時間は短くなる．このため，並列数が 5 の場合の処理時間は，並列数が 4 の場合に比べて短くなる．

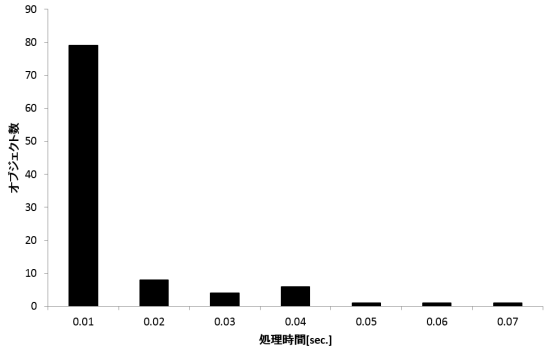

図 12 各オブジェクトにおけるボロノイ領域の作成にかかる処理時間の分布

7.5 オブジェクトごとの処理時間の分布

初期領域を考慮せずに CZ を用いる手法では，ボロノイ領域が無限遠まで到達するボロノイ辺をもつ場合，ボロノイ領域の処理時間は長大化する．そこで，空間上に存在するオブジェクト数が 100 の場合，提案手法でボロノイ領域の作成にかかる処理時間の分布を**図 12** に示す．横軸はボロノイ領域の作成にかかる処理時間の平均であり，縦軸はオブジェクト数である．

図 12 より，ボロノイ領域の作成にかかる処理時間に応じたオブジェクト数に偏りがある．並列処理を行う場合，ボロノイ領域の作成にかかる処理時間が長大化するスレッドが発生すると，このスレッドの処理が終了するまで他のスレッドは待機するため待ち時間が発生し，並列処理の効率は低下する．

次に，並列処理において各スレッドの待ち時間を確認するため，提案手法を用いて，5 つのスレッドが同時に動作して各スレッドがボロノイ領域の作成にかかる処理時間を**図 13** に示す．横軸はボロノイ領域の作成にかかる処理時間である．縦軸の thread1, ..., thread5 は各スレッドを表し，total はすべてのスレッドの処理が終了するまでにかかる処理時間を表す．空間上に存在するオブジェクト数は 100 とする．

図 13 より，thread5 の処理時間のみ長大化する．また，total の処理時間は 0.339 秒，thread5 の処理時間は 0.333 秒となり，thread5 の処理時間は他のスレッドに比べて長い．以上より，thread5 以外のスレッドで待ち時間が発生することが分かる．

7.6 スレッドの生成にかかる処理時間

評価環境では，選択したターゲットに対して，スレッドを生成するための 1 つのメインスレッド，およびボロノイ領域の計算に用いる 5 つのスレッドの計 6 スレッドが並列して動作する．このため，スレッドの生成，破棄，および切り替えで発生する処理時間を評価する必要がある．ボロノイ領域の作成で発生する平均処理時間を**図 14** に示す．

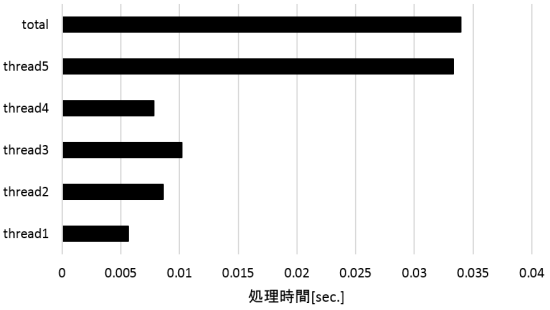

図 13 ボロノイ領域の作成にかかる各スレッドの処理時間

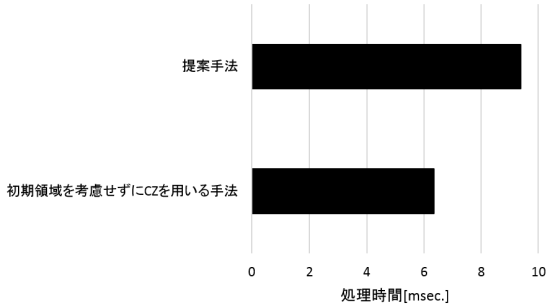

図 14 ボロノイ領域の作成にかかる平均処理時間

縦軸は評価手法であり，提案手法，および初期領域を考慮せずに CZ を用いる手法の二つとする．横軸はボロノイ領域の作成で発生する処理時間の平均である．空間上に存在するオブジェクト数は 100 とする．

図 14 より，提案手法でボロノイ領域の作成にかかる平均処理時間は約 9.37 ミリ秒となり，初期領域を考慮せずに CZ を用いる手法の平均処理時間である約 6.36 ミリ秒に比べて長大化する．提案手法では，各スレッドの処理時間が長大化したため，全体の処理時間が長大化した．提案手法，および初期領域を考慮せず CZ を用いる手法において，ボロノイ領域を作成する手順は同じである．このため，図 14 における処理時間の差は，スレッドの生成，破棄，および切り替え時で発生した処理時間の差である．

8. おわりに

本研究では，初期領域を考慮せずに CZ を用いる手法を並列化してボロノイ図作成にかかる処理時間を短縮する手法を提案した．提案手法では，初期領域を考慮せずに CZ を用いる手法をもとに，ターゲットごとに発生するボロノイ領域の作成処理を並列化することで，既存手法と比較して処理時間を短縮した．評価において，オブジェクト数 100，並列数 5 の場合，提案手法でボロノイ図の作成にかかる処理時間は，並列化を行わない既存手法と比べて処理時間を約 15.9%短縮した．

今後は，並列数を増加させた場合の評価，および MapReduce を用いた評価が考えられる．

謝辞 本研究の一部は，文部科学省科学研究費補助金・若手研究（B）(26730059)，基盤研究（B）(15H02702)，基盤研究（C）(16K01065)，および（公財）山陽放送学術文化財団の研究助成による成果である．ここに記して謝意を表す．

参考文献

[1] Okabe, A., Boots, B., Sugihara, K., and Chiu, S. : *Spatial Tessellations Concepts and Applications of Voronoi Diagrams, 2nd Edition*, John Wiley Sons, Chechester, England (2000).

[2] Yang, S., Cheema, M., A., Lin, X., and Wang, W. : *Reverse k Nearest Neighbors Query Processing: Experiments and Analysis*, Proceeding of VLDB Endowment, Vol.8, Issue 5, pp.605-616 (2015).

[3] Costello, B. : *Calculating Voronoi Diagrams Using Simple Chemical Reactions*, Proceeding of Parallel Processing Letters, Vol.25, Issue 1, pp.1-26 (2015).

[4] Guttman, A. : *R-trees: A Dynamic Index Structure for Spatial Searching*, Proceedings of 1984 ACM SIGMOD International Conference on Management of Data, Vol.14, Issue 2, pp.47-57 (1984).

[5] Gionis, A., Indyk, P., and Motwani, R. : *Similarity Search in High Dimensions via Hashing*, Proceeding of the 25th International Conference on Very Large Data Bases, pp.518-529 (1999).

[6] Ciaccia, P., Patella, M., and Zezula, P. : *M-tree: An Efficient Access Method for Similarity Search in Metric Spaces*, Proceedings of the 23rd International Conference on Very Large Data Bases, pp.426-435 (1997).

[7] Wu, W., Yang, F., Chan, C., Y., and Tan, K., L. : *FINCH: Evaluating Reverse k-NearestNeighbor Queries on Locational Data*, Proceedings of VLDB Endowment, Vol.1, Issue 1, pp.1056-1067 (2008).

[8] Adhinugraha, K., M., Taniar, D., and Indrawan, M. : *Finding Reverse Nearest Neighbors by Region, Concurrency and Computation*, Practice and Experience, Vol.26, Issue 5, pp.1142-1156 (2014).

[9] Green, P., J., and Sibson, R. : *Computing Dirichlet Tessellations in the Plane*, Proceeding of Computer Journal, Vol.21, Issue 2, pp.168-173 (1978).

[10] 河野 洋一, 西松 研, 福盛 秀雄, 村岡 洋一 : 有限要素法要素分割の並列化 -ボロノイ分割の並列化-, 情報処理学会研究報告, Vol.1995-HPC-60, No.22, pp.43-48 (1996).

[11] Dean, J., and Ghemawat, S. : *MapReduce: simplified data processing on large clusters*, Proceeding of Communications of the ACM, Vol.51, Issue 2, pp.107-113 (2008).