移動軌跡の交点を用いたジオソーシャルデータに対するクラスタリング手法

伊藤光太郎† 横山 昌平††

† 首都大学東京 システムデザイン学部〒 191–0065 東京都日野市旭が丘 6-6
†† 首都大学東京大学院 システムデザイン研究科〒 191–0065 東京都日野市旭が丘 6-6
E-mail: †ito-kotaro@ed.tmu.ac.jp, ††shohei@tmu.ac.jp

あらまし 近年，Twitter や Instagram などの SNS が広く普及し，それに付随してインターネット上に多くのジオタグ付きのデータが存在するようになった．しかし，データの数が膨大になったため，重要なデータをクラスタリングし，抽出することの必要性が高まった．地理的なクラスタリングアルゴリズムの一つに「EBSCAN」というものがある．このアルゴリズムは移動軌跡の交点に着目したクラスタリング手法で，点群をクラスタリングする「DBSCAN」とは異なり，実際の地形情報を考慮に入れたクラスタリングをすることができる．本論文ではこの EBSCAN を参考に，交点探索を Bentley-Ottman アルゴリズムや入力線分の分割処理を用いて行うことで，処理時間が短縮されたクラスタリング手法を提案する．提案手法ではデータの緯度経度および時間から軌跡を計算し，その軌跡の交点を求め，その交点と元データの距離の値を用いてクラスタリングを行う．本研究では写真共有サイトである Flickr からデータを取得し，提案手法と DBSCAN とのクラスタリングにかかる処理時間，およびクラスタリング結果を比較した．本研究では，使用するデータの軌跡を単純な計算によって求めたものを使用したため，より現実の移動に即したデータを用いてクラスタリングをすることが今後の課題である．

キーワード クラスタリング, Flickr, ジオソーシャルデータ, Bentley-Ottman アルゴリズム, 並列処理

1 はじめに

近年，Twitter[1] や Instagram[2]，Flickr[3] をはじめとした SNS が広く普及したことにより，多くの人々が気軽にテキストデータや写真，動画などのマルチメディアデータを投稿するようになった．Twitter,Inc. が 2019 年 10 月 24 日に発表した資料 [1] によると，2019 年 7 月から 2019 年 9 月における全世界での 1 日当たりのアクティブユーザ数は 1 億 4500 万人を超えている．

また，旅行の事前情報として SNS を利用する人の割合も増加傾向にある．観光庁が実施している訪日外国人消費動向調査では，2018 年の年次報告書 [2] における「出発前に得た旅行情報源で役に立ったもの」という質問に対して「SNS」が 23.7%であり，2 番目に多い回答となっていた．このことから今後，旅行先で写真を撮影しインターネット上にアップロードされる件数が増加していくこと，および旅行の参考にするための情報源として SNS 上のデータが用いられる機会も増加していくことが予想される．

さらに，Microsoft が 2019 年 7 月 25 日に発表した IoT Signals [3] によると，世界の大企業の 84%が IoT ソリューションを導入しており，2021 年にはその割合が 94%まで上昇すると予想されている．IoT の中にはスマートタグや GPS トラッカーなど，位置情報を組み込んだシステムも多く存在する．そのため，今後は人間がアップロードするデータ以外にも，地理的な情報をもつデータが増加していくと考えられる．

以上より，現代においてはインターネット上に多くの地理的な情報を含むデータが存在していることが確認でき，今後もこのようなデータは増加していくと予測する．その中で，地理的なデータを有効活用することがより重要になると考えられる．

しかし，データの数が膨大になったため，手動でデータを抽出し活用するのは莫大な労力がかかる．従って，データをクラスタリングし，重要なデータを抽出することが必要である．

SNS データの分析の例として，Sakaki ら [4] の研究のように場所に注目した分析が多いことが挙げられる．このことから重要なデータ群の例として,「注目度の高い領域のデータ群」というものがあると考えられる．注目度の高い領域を発見することで，人々が注目するスポットやイベントの発生の検出が容易となる．

ここで，注目度の高い領域とはどのような領域かを考える．本研究では「その領域内で多くのデータが存在する」という領域，すなわちデータの分布が高密度である領域を注目度の高い領域として定義する．高密度の領域を発見するためによく使われるアルゴリズムとして，Ester ら [5] が提案した密度ベースのクラスタリングを行う DBSCAN が存在する．DBSCAN は近傍の点を用いて密度を計算し，その密度が閾値以上ならばクラスタに追加するアルゴリズムである．

しかし，DBSCAN ではクラスタリングの際に近傍距離 ε とその半径内に必要とする点の個数 $MinPts$ という 2 つのパラメータを設定する必要があり，データに対する深い事前知識が

1 : https://twitter.com/
2 : https://www.instagram.com/
3 : https://www.flickr.com/

ない限り，理想的なパラメータの値を設定するのが困難である．特に，ソーシャルデータに関しては全体のデータの粗密が不明であることが多く，クラスタリングを行う前に適切なパラメータを調査する必要があることが多い．また，DBSCAN では入力データとして点群を用いるため,「距離は近いが，実際の移動は困難である二点」を同一クラスタに振り分けてしまうことがある．

これらの問題を解決するためのアルゴリズムとして，Yokoyama ら [6] の EBSCAN というクラスタリングアルゴリズムがある．これは移動軌跡の絡まりに着目することによって，実際の移動を考慮に入れたアルゴリズムとなっている．さらに DBSCAN ではパラメータが密度に紐づいた 2 つの値であったが，EBSCAN では設定が必要なパラメータは距離に関する 1 つの値のみとなっている．そのため，密度を定義するよりもパラメータの設定が容易であり，使用者にとって使いやすいアルゴリズムであると考えられる．

本研究ではこの EBSCAN を参考に，インターネット上にあるジオタグ付きのデータから移動軌跡を計算し，それらの交点を用いてクラスタリングを行う手法を提案する．本研究の EBSCAN を発展させた主な部分は，交点探索の際に使用するアルゴリズムを変更した点と，その際に入力データの分割をすることで並列処理を行う点である．並列処理は従来の EBSCAN よりも大量のデータを高速に処理することを目的とした．

本論文の構成は以下のとおりである．2 章では，関連研究を述べる．3 章では，提案手法を述べる．4 章では提案手法と DBSCAN を用いた実験結果を述べ，5 章で，両手法を比較し考察を述べる．6 章で本研究のまとめと今後の課題を述べる．

2 関 連 研 究

密度ベースのクラスタリングに関する研究は様々なものが存在する．

最も有名なものの 1 つとして，1 章で述べた Ester らが提案した DBSCAN があげられる．これは事前に近傍半径 ε と最小点数 $MinPts$ を設定し，各点の近傍の密度を計算する．近傍半径 ε 以内に $MinPts$ 以上の点が存在する場合はその点をコア点とし，コア点ではない点の中でコア点から ε 以内にある点を到達可能点，そうでない点を外れ値に分類し，コア点および到達可能点でクラスタを構成するアルゴリズムである．

Nasibov ら [7] はこの DBSCAN の密度の計算を行う際に，基準となる点から近傍半径 ε 内に存在する点を一意にカウントするのではなく，関数を用いることで離れている距離によって異なるスコアをつけるアルゴリズムを提案した．これによって，DBSCAN よりも異なる形状や密度のデータセットに対して頑強な結果を得られることを示した．

Dash ら [8] は DBSCAN と重心ベースのクラスタリングアルゴリズムである k-means の 2 つのアルゴリズムを用いることで，k-means の利点である高速性と DBSCAN の利点である頑健性を併せ持つ手法を提案した．また，この手法では k-means を DBSCAN よりも事前に行うことで，DBSCAN における MinPts のパラメータの設定を容易なものとしている．

また，ソーシャルネットワーク上のデータに関するクラスタリングアルゴリズムの研究も多岐にわたる．

Shi ら [9] はジオソーシャルネットワークのユーザが訪れる場所を密度ベースでクラスタリングするために，ユークリッド距離を正規化した空間距離とユーザ間の友人関係を考慮した社会的距離の両方を考慮したジオソーシャル距離を用いて密度計算を行った．これにより，社会的に高品質なクラスタを発見することを可能とした．

Kisilevich ら [10] はジオタグ付き写真のコレクションに関して，データの分布が高密度な領域を発見する P-DBSCAN を提案した．これは密度を計算する際に，写真の点数ではなく，その近傍に存在する写真を撮影したユーザ数をパラメータとした点や，コア点に関しての密度の変化を考えることで急激に密度が変化した場所を検出を可能とした点が DBSCAN と異なる．

さらに，先行研究として Yokoyama らの EBSCAN がある．これは，入力データからユーザの移動軌跡を考えることで，DBSCAN よりもパラメータの数を減少させつつ，地理的な特徴に応じたクラスタリングを可能とさせるアルゴリズムである．本研究では上記の研究と同様にジオタグ付き写真のコレクションを入力データセットとして使用する．

本研究と先行研究の差異は，交点を探索する際のアルゴリズムとして，先行研究では総当たりもしくは R-tree を用いて実装していたが，本研究では Bentley ら [11] が提案したアルゴリズムを用いることである．また，本研究では移動軌跡の交点を探索する際に領域を分割し，並列処理による処理時間の短縮を目指したことも独自の点である．

3 提 案 手 法

図 1 に提案手法の概要を載せる．提案手法の手順は大きく分けて三種類のフェイズから構成される．

はじめに，軌跡計算フェイズにてインターネット上に存在する緯度経度情報を含んだ写真に関するデータを取得し，そのデータの移動軌跡を計算する．ここで，写真の撮影位置は計算された移動軌跡の端点となる．

次に，移動軌跡の座標を緯度経度で表す地理座標系から x, y で表す直行座標系に置き換え，交点探索フェイズで入力された移動軌跡の交点探索を行う．

最後に，発見された交点の座標を地理座標系に戻し，クラスタリングフェイズを実行する．クラスタリングフェイズでは，まず交点を有する線分の組み合わせに対して端点間の距離を計算する．その計算された距離が事前に設定された閾値未満である場合，端点 (写真) のクラスタリングをおこなう．

3.1 データの取得

本研究では Flickr API を用いて，Flickr に投稿された画像のデータを JSON 形式で取得する．ここで取得するデータにはその画像固有の photoID，その画像を投稿したユーザ ID，その画像を撮影した緯度経度，および撮影日時などの情報が含ま

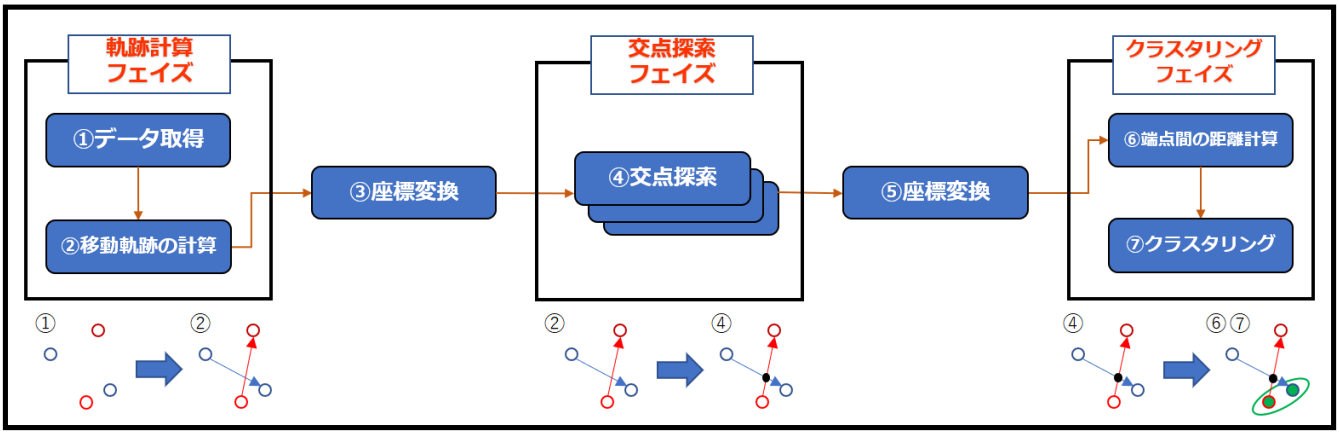

図 1　提案手法の流れ

れている．

また，本研究では緯度経度を指定して Flickr から画像データを取得したが，Flickr API の仕様上，一度のクエリで取得できるデータ数は 4000 件程度である．そのため，本来取得したい領域を複数に分割し，クエリの回数を増やすことで指定した領域内の全データを取得できるようにした．

3.2　移動軌跡の計算

次に，取得したデータを用いて移動軌跡の計算をおこなう．

まず，photoID の重複の有無をチェックし，すべての画像がユニークとなるように重複画像を消去する．

次に，ユーザ ID を頼りに全ての画像を撮影ユーザごとにまとめる．さらに，各ユーザ内で写真を旅行単位でまとめる．各ユーザにおいて撮影した写真を撮影時刻順で並び替え，連続する 2 つの写真の撮影時刻が離れすぎていた場合，そこを区切りに別の旅行として扱う．本研究では連続する写真が 12 時間以上離れていた場合，別の旅行として扱った．旅行単位でまとめた写真のうち，写真が 1 枚しかないものに関しては軌跡を考えることができないため，その写真は消去する．

そして，全ての旅行の写真をまとめ，撮影時刻順にチェックしていく．同じ旅行に属する写真が出現した際に，その写真とその写真の前に出てきた同一旅行に属する写真とで軌跡を生成し，生成した軌跡に軌跡 ID をつける．この手順により，撮影時刻が古いものから軌跡を生成していくことが可能となる．

最後に，軌跡の二端点が同じ座標の場合はその軌跡を消去する．Flickr から取得されたデータのうち，一部のユーザでは撮影した全ての画像の緯度経度情報に同じ座標が入っている場合がある．ここでの軌跡消去は，そういった正しくないと思われる緯度経度情報を含むデータを除外することが目的である．

また，本研究ではユーザの移動軌跡が重なるような場合は軌跡を一つにまとめて考えるため，重複する移動軌跡が存在する場合は片方の軌跡を除去する．

3.3　座 標 変 換

本研究では交点探索を行う前後に座標変換を行う．座標変換を行う理由としては，Flickr API で取得した緯度経度情報は小数点第 6 位の精度までの情報が含まれているが，本研究では指定した領域の範囲が 1 度未満なため，小数点第 6 位までのデータでは浮動小数点由来の計算誤差が含まれやすいと考えた．そのため，以下の式で地理座標系から直行座標系への変換をおこなった．ここで，lon，lat はそれぞれ本来の経度と緯度，$minlon$，$minlat$ は指定した領域の最小の経度と緯度である．

$$x = lon - minlon * 10^9$$

$$y = lat - minlat * 10^9$$

この状態で 3.4 節の交点探索を行った後に，逆変換の式を用いることで，地理座標系へ戻した上で，3.5 節における距離計算と 3.6 節のクラスタリングをおこなった．

3.4　交 点 探 索

交点探索は Bentley らが提案したアルゴリズムをベースにしておこなう．これは入力線分に関するすべての交点を求めるためのアルゴリズムである．

アルゴリズムの考え方としては，走査法のように掃引線が x 軸上を一定方向に移動していく．この掃引線が線分の端点もしくは交点を発見した場合に，その点の種類に応じてイベントを起こす．このイベントの種類によって，現在掃引線がある x 軸上での各線分の y 値の大小関係，すなわち位相を更新する．この位相を管理することで，イベントに関係する点の線分の一つ上の線分や一つ下の線分との交点の有無のみを計算する．従って，遠く離れた線分との交点を探索せずに，効率の良い交点探索が行える．

イベントは以下の通りである．

(1) 点が線分 l_1 の左端点

(2) 点が線分 l_1 の右端点

(3) 点が線分 l_1 と線分 l_2 の交点

また，実装上の問題により，本研究では以下のイベントを追加することで，複数の線分が一点で交わる場合を考慮した．

(4) 点が 3 線分以上の交点

ここで，入力する情報は移動軌跡の右端点および左端点であり，出力される情報は移動軌跡の交点である．また，出力される交点はその位置情報および，その交点を構成する二つの移動軌跡の軌跡 ID の情報を持つ．

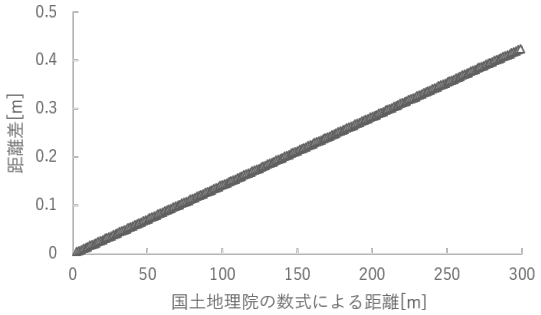

図 2　ヒュベニの簡略式の絶対誤差

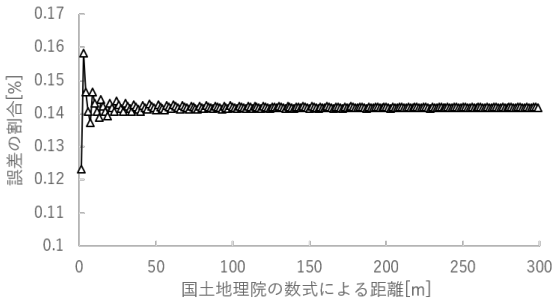

図 3　ヒュベニの簡略式の相対誤差

本研究では，この交点探索を行う前のタイミングで入力データの分割を行う．まず，入力データの端点を x 値，y 値の順番で優先度をつけ，ソートする．その端点を順番にチェックしていき，一定数の右端点がチェック済みになった時点での x 値を基準に，領域を分割する．領域の分割時に，左端点はチェック済みであるが右端点が未出現であるような線分が存在する場合，領域が分割された x 値の時点でのその線分の y 値を計算し，その値を右端点として直前の領域に挿入する．また，この y 値を分割した次の領域に左端点として挿入する．これを繰り返し，全ての線分の端点がチェック済みとなるまで続ける．

なお，本論文では分割したデータを並列処理した場合に関しても処理時間の計測を行った．本研究での並列処理では Python の concurrent.futures モジュールを用いて実装し，並列数は 4 に設定した．

3.5　端点間の距離計算

ここでは，発見した交点に関係する線分の端点の距離計算を行う．交点を構成する線分をリストから取り出し，その線分の端点の情報を調べる．その端点の 4 つの組み合わせに関して，二点間の距離を計算する．本研究ではヒュベニの簡略式と呼ばれる，Hubeny の厳密解の項数第一項の式を用いて緯度経度から距離計算を行った．

ヒュベニの簡略式は以下の数式で表される．

$$d = \sqrt{(d_y M)^2 + (d_x N cos\ \mu_y)^2}$$

ここで，M は子午線曲率半径，N は卯酉線曲率半径，μ_y は二点の緯度の平均値，d_y,d_x はそれぞれ二点の緯度と経度の差である．また，本研究において，子午線曲率半径と卯酉線曲率半径の計算には世界測地系の GRS80 楕円体の長半径と短半径の値を用いた．

本研究では，この簡略式が実用的な計算精度を持つかどうかの確認として，国土地理院が提供している測量計算サイト [4] を用いて比較を行った．測量計算サイトでは Bowring [12] の文献を参考とした数式を用いて距離計算を実装している．比較の方法としては，出発点を北緯 36 度，東経 138 度に設定し，緯度経度をそれぞれ 0.00001 度ずつずらしていった点を到着点とした際のヒュベニの簡略式での値と測量計算サイトでの値を比較した．比較した結果を図 2，図 3 に示す．

4：https://vldb.gsi.go.jp/sokuchi/surveycalc/surveycalc/bl2stf.html

図 2 を見ると，ヒュベニの簡略式による値と国土地理院の数式による値の絶対誤差は二点間の距離に比例して増加する傾向がある．また，図 3 を見ると，ヒュベニの簡略式の値と国土地理院の数式による値の相対誤差は最も高い場合でも 0.16%程度であり，二点間の距離が増加するにつれて，0.14%程度に収束していった．

本論文では，クラスタリング対象を撮影された写真に限定しているため，提案手法でのパラメータである $toofar$ や DBSCAN でのパラメータである ε の値として $300m$ を超えるケースは少ないと考えられる．従って，本論文で想定される値の範囲内では，ヒュベニの簡略式は実用的な計算精度を持つと言える．

ここで，二点間の距離に関するパラメータ $toofar$ を設定する．ヒュベニの簡略式を用いて距離計算を行った結果，その距離が $toofar$ 以下であれば，リストに追加する．

3.6　クラスタリング

3.5 節で作成したリストから二端点を取得し，それぞれの点がいずれかのクラスタに属しているかどうかを調べ，所属状況に応じて以下の方法でクラスタリングを行う．クラスタリングを行う順番としては，交点が生成された時期が早いもの，つまり，その交点を形成する二つの軌跡の軌跡 ID のうち大きいものを基準として優先度をつける．なお，交点の生成時期が同じものに関しては，x 値が小さいものを優先とし，x 値も同じ場合は y 値が小さいものを優先とした．

(1) どちらの端点もクラスタに属していない場合
新たにクラスタを作り，そのクラスタに 2 つの端点を追加する．所属済み端点リストにその二端点の番号を所属先クラスタ番号とともに追加する．

(2) 片方の端点のみクラスタに属している場合
どのクラスタにも所属していない端点を，もう片方の端点が所属しているクラスタに追加する．所属済み端点リストに追加した端点の番号を所属先クラスタ番号とともに追加する．

(3) 両方の端点がクラスタに属している場合
そのクラスタに追加された際の距離 (既存距離) と，現在の二端点での距離を比較し，現在の距離の方が短ければ，クラスタの更新を行う．

(3-a) どちらか一方の既存距離が現在の距離よりも長い場合
既存距離が長い方のクラスタからその要素を除去し，既存距離が短い方のクラスタにその要素を追加する．また，移動させた

端点の所属済み端点リストにあるクラスタ番号を追加した方のクラスタ番号に書き換える．

(3-b) 両方の既存距離が現在の距離よりも長い場合
2 つのクラスタを結合し，関係する所属済み端点リストの情報を書き換える．

(3-c) 両方の既存距離が現在の距離よりも短い場合
クラスタに関する変更は行わない．

これを繰り返し，最終的にリストにある交点をすべて処理する．その後，要素の移動によりクラスタ内の要素数が 1 以下になってしまったクラスタを除去し，その結果を最終的なクラスタとして出力する．なお, ここ要素数が 1 になったクラスタの要素以外に，他の軌跡と交点を持たない軌跡の端点や，他の軌跡との交点を持つが，その軌跡の二端点との距離が $toofar$ よりも大きいような端点は外れ値として，どのクラスタにも属さない点となる．

4 実　　験

本研究では，入力データとして FlickrAPI から取得したデータを用いる．使用するデータはデータ 1 が東経 139.475 度から東経 139.485 度，北緯 35.298 度から北緯 35.302 度でのデータ 750 件，データ 2 が東経 135.700 度から東経 135.800 度，北緯 34.970 度から北緯 35.070 度でのデータ 2500 件，データ 3 が東経 138.980 度から東経 139.050 度，北緯 35.180 度から北緯 35.250 度でのデータ 6551 件である．

これらのデータに対して，DBSCAN と提案手法でクラスタリングを行い，それぞれの処理時間および出力されたクラスタを比較する．まず，複数の方法で実装した交点探索にかかる処理時間の比較を表 1 に示す．ここで，B&O は Bentley-Ottman アルゴリズムを用いて実装した場合の交点探索を示す．

表 1　交点探索にかかる時間

	データ 1[秒]	データ 2[秒]	データ 3[秒]
B&O(分割なし)	0.601862	1.205643	24.016502
B&O(分割あり)	0.572611	0.957345	10.776461
B&O(並列処理)	2.863002	3.055112	6.974760
総当たり	2.250974	14.067048	139.678907

次に提案手法における，各段階での処理時間を表 2 に示す．ここで，交点探索は分割処理を行った場合での Bentley-Ottman アルゴリズムを用いて計測された処理時間を記載し，クラスタリングフェイズは $toofar = 35m$ に設定した際の処理時間を記載した．

表 2　提案手法の各段階における処理時間

	データ 1[秒]	データ 2[秒]	データ 3[秒]
移動軌跡の作成	0.008003	0.027962	0.140613
座標変換	0.003997	0.007025	0.015609
交点探索	0.572611	0.957345	10.776461
座標変換	0.003994	0.007988	0.093726
距離計算	0.026644	0.031284	0.374913
クラスタリング	0.003999	0.004979	0.032767

続いて，R-tree を用いて実装をおこなった DBSCAN の各段階での処理時間を表 3 に示す．ここで，クラスタリングフェイズは $\varepsilon = 12.5m$，$MinPts = 5$ に設定した際の処理時間を記載した．

表 3　DBSCAN の各段階における処理時間

	データ 1[秒]	データ 2[秒]	データ 3[秒]
R-tree 構築	0.015613	0.046851	0.235718
クラスタリング	0.109349	0.140573	0.919548

次に，図 1 における提案手法でのクラスタリングフェイズにかかる処理時間と DBSCAN の処理時間との比較を表 4 に示す．ここで，提案手法の数値は表 2 での距離計算とクラスタリングの数値の和を記載した．

表 4　クラスタリングの処理時間の比較

	提案手法 [秒]	DBSCAN[秒]
データ 1	0.030643	0.109349
データ 2	0.036263	0.140573
データ 3	0.407680	0.919548

最後に，各手法で作成されたクラスタを実際の地図上にマッピングした結果を示す．図 4 が提案手法によって作成されたクラスタをマッピングしたもので，図 5 および図 6 が DBSCAN によって作成されたクラスタをマッピングしたものである．それぞれの図において，同一のクラスタに属するデータは同一のマーカーによってマッピングした．なお，それぞれの手法でどのクラスタにも属さなかったデータに関しては，マーカーによるマッピングは行わなかった．

ここで，図 4 のパラメータ設定は $toofar = 35m$，図 5 のパラメータ設定は $\varepsilon = 12.5m$，$MinPts = 5$，図 6 のパラメータ設定は $\varepsilon = 17.5m$，$MinPts = 5$ とした．

5 考　　察

本章では 4 章の実験に対して，処理時間およびクラスタリング結果に対する考察を行う．

5.1 処 理 時 間

まず，交点探索の処理時間について比較する．

表 1 より，データの分割処理を行った場合と，そうでない場合を比較すると，データの分割処理を行った方が処理時間が短くなるという結果が出た．これは，データの分割を行うことによって，線分を探索するのに必要とする時間が短くなったためであると考えられる．

並列処理に関してはデータ 1 の時点では最も処理時間が大きいが，データ 2 では総当たりよりも処理時間が小さくなり，データ 3 では最も小さな処理時間となった．これは，並列処理を行うプロセスの起動には一定の時間を要するため，データ数の少ないデータ 1 やデータ 2 の時点ではプロセス起動の時間が実際に交点探索を行う部分よりも長く，全体の処理時間も大きくなっていたと考えられる．しかし，データ数が増加したデー

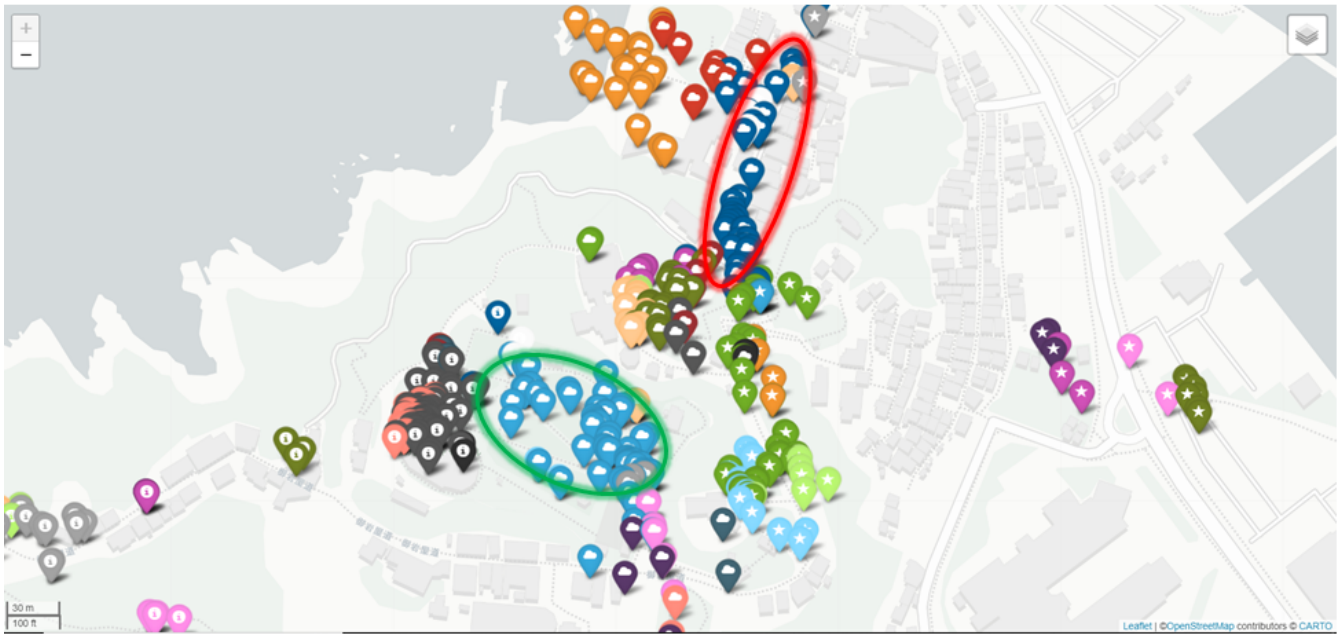

図 4　データ 1 に対する提案手法 ($toofar = 35m$) のクラスタリング結果

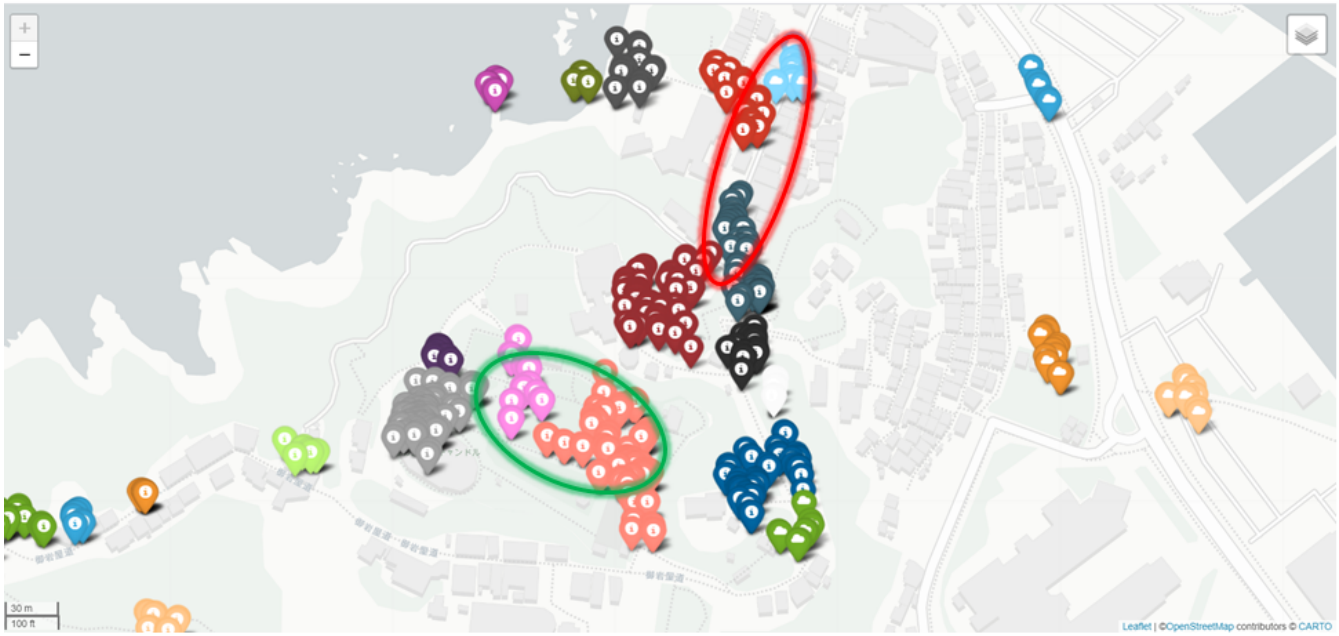

図 5　データ 1 に対する DBSCAN($\varepsilon = 12.5m$, $MinPts = 5$) のクラスタリング結果

タ 3 ではプロセス起動の時間が占める割合が低下したため，他の方法よりも小さな処理時間での交点探索が可能となったのだと考えられる．

また，全体の処理時間について考えると，表 2 および表 3 より，提案手法の移動軌跡の作成からクラスタリングまでの処理時間の合計と DBSCAN の合計の処理時間を比較した場合，いずれのデータでも DBSCAN の方が小さな値となっている．しかし表 4 のように，クラスタリングフェイズに含まれるプロセスのみの処理時間を考えた場合，提案手法の方が DBSCAN よりも短い処理時間でクラスタリングが可能となっている．

提案手法ではデータの追加や削除といった変更が起こらない限り，パラメータを変えて再びクラスタリングを行う際にはクラスタリングフェイズの部分のみを再計算すれば良い．従って，DBSCAN よりも適切なパラメータを求めるのに必要とする時間が短くなるという利点があると言える．

5.2　クラスタリング結果

次に，クラスタリング結果について比較する．

図 4 から図 6 では同一地点に赤の楕円と緑の楕円で囲んだ領域がある．この領域はそれぞれ「弁天財仲見世通り」と呼ばれる商店街と「江の島サムエル・コッキング苑」という観光施設が存在する領域である．この 2 つの領域に関して比較する．

図 4 より，提案手法では赤色の楕円内部をほぼ青色のマーカー一種類が占めていることに加え，赤色の楕円外部には青色

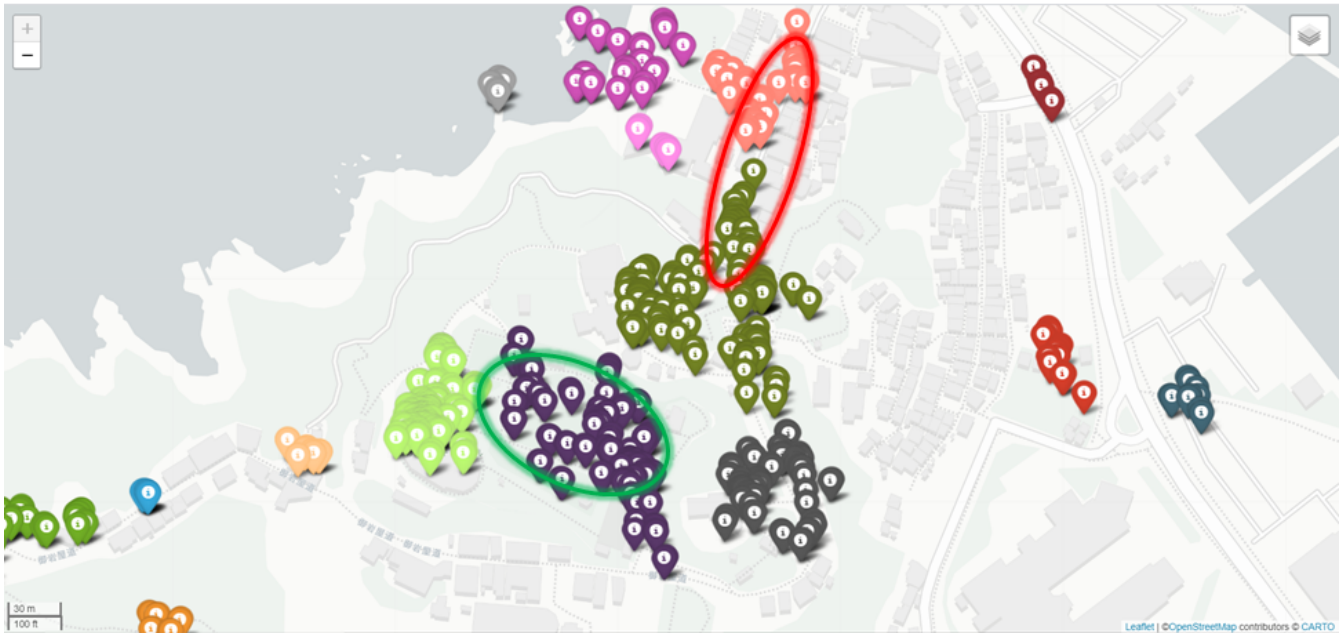

図 6　データ 1 に対する DBSCAN($\varepsilon = 17.5m$，$MinPts = 5$) のクラスタリング結果

のマーカーが大きくはみ出していないことが確認できる．また緑色の楕円でも同様に，水色のマーカーが楕円内部の大部分を占め，緑色の楕円外部には水色のマーカーが少ない．これらのことから，提案手法では各領域を一つのクラスタとして抽出できたと言える．

続いて，DBSCAN を用いた場合のクラスタについて確認する．図 5 では，どちらの楕円においても楕円内を占めているマーカーの楕円外部へのはみ出しは少ない．しかし，赤色の楕円部の内部には主に三種類の色のマーカーが出力されており，緑色の楕円内部では主に二種類の色のマーカーが出力されているため，それぞれの領域を 1 つのクラスタとして取得できているとは言い難い状況となっている．

また，図 6 を見ると緑色の楕円では紫色のマーカーが広域にわたるクラスタを作っており，外部へのはみ出しも比較的小さいため，江の島サムエル・コッキング苑に関しては 1 つのクラスタで表すことに成功していると言える．しかし，赤い楕円に関しては主に二種類のマーカーが出力されており，一つのクラスタとして抽出できていないことに加え，赤色の楕円内部の南側を占めるアイコンマーカーが楕円を大きくはみ出し，他の観光スポットが存在するエリアをも覆ってしまっていることがわかる．このことから，弁天財仲見世通りに関しては 1 つのクラスタで表せていない状況であると言える．

これらの違いについて考察する．弁天財仲見世通りに関しては，終端部分で道が分岐し，軌跡が分散することに加え，観光スポットの移動間では写真を撮影する人が少ないと考えられる．つまり，商店街内部で撮影した二枚の写真は撮影地点間の距離が小さくなるが，商店街内部で撮影した写真と他の観光スポットで撮影した写真は撮影地点間の距離が遠くなる．そのため，観光スポットをまたぐような移動軌跡に関する写真の組み合わせは撮影地点間の距離が $toofar$ よりも大きな値となってしまい，別のクラスタとして出力されたのだと考えられる．

江の島サムエル・コッキング苑に関しては，入り組んだ道であるが，その通路内で写真を撮影する人が多かったため，移動軌跡の交点が存在しやすかったと考えられる．その結果，DBSCAN では別のクラスタとして出力されるようなクラスタが提案手法では結合し，出力されたのだと考えられる．

このことから，本研究の実験において，提案手法は観光スポットごとのクラスタを出力することに一定の成果を挙げ，提案手法は人の移動に沿ったクラスタを発見することに対して有効であると考えられる．

6　まとめと今後の課題

本研究では，EBSCAN を参考に移動軌跡の交点に着目したクラスタリングアルゴリズムを提案した．提案手法ではクラスタリングフェイズにおけるパラメータの個数が DBSCAN よりも少なく，データ間の距離のみをユーザが指定すれば良い．従って，データの密度を想定してパラメータ設定しなければならない DBSCAN よりも，使用しやすいアルゴリズムであると考えられる．

また，同一データに対してパラメータを変更しクラスタリングをおこなう場合，DBSCAN ではすべての計算を再び行う必要があるが，提案手法では，クラスタリングフェイズのみをやり直せば良いので，結果的に適切なパラメータを定めるまでにかかる時間が短くて済むと考えられる．

加えて，本論文では移動軌跡の交点を探索する際の手法として Bentley らのアルゴリズムを用い，データの並列処理を行った．これにより，総当たり探索よりも高速に交点の発見が可能となり，結果的に全体の処理速度の向上にもつながった．

さらに，移動軌跡に着目したことによって，通路が入り組んでいながらも，一か所である観光スポットを 1 つのクラスタとして検出し，商店街と他の観光スポットを別のクラスタとして

抽出できたことから，提案手法は実際の人の移動に即したクラスタを発見するのに有効であることを示した．

今後の課題としては，本研究では「移動軌跡の計算を入力データを直線で結ぶ」という単純な計算によって生成したものを用いたため，実際の移動情報とはずれがある．そのため，より正確な移動経路の情報を含むデータを用いて実験を行う必要があると考えられる．また，提案手法において，交点探索に要する処理時間が全体の大部分を占めている．そのため，より良いデータ構造を用いて実装をすることで，交点探索に必要な処理時間を短縮し，全体の処理時間をより短くすることも必要であると考える．

文　献

[1] Twitter,Inc. Q3 2019 letter to shareholders. https://s22.q4cdn.com/826641620/files/doc_financials/2019/q3/Q3-2019-Shareholder-Letter.pdf.

[2] 国土交通省観光庁. 2018 年 年次報告書. http://www.mlit.go.jp/common/001285944.pdf.

[3] Microsoft. Iot signals summary of research learnings 2019. https://azure.microsoft.com/mediahandler/files/resourcefiles/iot-signals/IoT-Signals-Microsoft-072019.pdf.

[4] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pp. 851–860. ACM, 2010.

[5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, Vol. 96, pp. 226–231, 1996.

[6] Shohei Yokoyama, Ágnes Bogárdi-Mészöly, and Hiroshi Ishikawa. Ebscan: An entanglement-based algorithm for discovering dense regions in large geo-social data streams with noise. In *Proceedings of the 8th ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, p. 7. ACM, 2015.

[7] Efendi N Nasibov and Gözde Ulutagay. Robustness of density-based clustering methods with various neighborhood relations. *Fuzzy Sets and Systems*, Vol. 160, No. 24, pp. 3601–3615, 2009.

[8] Manoranjan Dash, Huan Liu, and Xiaowei Xu. '1 + 1 > 2': merging distance and density based clustering. In *Proceedings Seventh International Conference on Database Systems for Advanced Applications. DASFAA 2001*, pp. 32–39. IEEE, 2001.

[9] Jieming Shi, Nikos Mamoulis, Dingming Wu, and David W Cheung. Density-based place clustering in geo-social networks. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 99–110. ACM, 2014.

[10] Slava Kisilevich, Florian Mansmann, and Daniel Keim. P-dbscan: a density based clustering algorithm for exploration and analysis of attractive areas using collections of geotagged photos. In *Proceedings of the 1st international conference and exhibition on computing for geospatial research & application*, p. 38. ACM, 2010.

[11] Jon Louis Bentley and Thomas A Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on computers*, No. 9, pp. 643–647, 1979.

[12] BR Bowring. Total inverse solutions for the geodesic and great elliptic. *Survey Review*, Vol. 33, No. 261, pp. 461–476, 1996.