情報処理学会論文誌　データベース　Vol.14　No.3　1–15 (July 2021)

移動軌跡の交点を用いた密度クラスタリングアルゴリズム

伊藤 光太郎[1,a)]　横山 昌平[1,b)]

受付日 2020年12月4日, 採録日 2021年4月13日

概要：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，分類することの必要性が高まった．我々は地理的な密度クラスタリングアルゴリズムの基礎的な考え方として「EBSCAN」を提案した．これは移動軌跡の交点に着目した考え方であり，実際の地形情報を考慮に入れた密度クラスタリングができること，および設定するパラメータの数が少ないことを利点とする．本論文では効率的な Bentley-Ottman アルゴリズムを実装し，DBSCAN に対する EBSCAN の有用性を示す．加えて，Bentley-Ottman アルゴリズムを用いるうえでの計算精度に起因する問題点と，それに対する解決方針を示す．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の質と処理時間を比較した．

キーワード：密度クラスタリング，ジオソーシャルデータ，Bentley-Ottman アルゴリズム，移動軌跡

Density Clustering Method Using Intersections of Trajectories

Kotaro Ito[1,a)]　Shohei Yokoyama[1,b)]

Received: December 4, 2020, Accepted: April 13, 2021

Abstract: In recent years, SNS such as Twitter and Instagram have become widespread, and along with this, a lot of geotagged data has come to exist on the Internet. However, as the number of data has become enormous, the need to cluster and classify important data has increased. We proposed “EBSCAN” as the basic idea of the geographical density clustering algorithm. This is a concept that focuses on the intersections of movement loci, and has the advantages of being able to perform density clustering that takes into account actual topographical information and having a small number of parameters to be set. In this paper, we implement an efficient Bentley-Ottman algorithm and show the usefulness of EBSCAN for DBSCAN. In addition, the problems caused by the calculation accuracy in using the Bentley-Ottman algorithm and the solution policy for them are shown. In this study, we obtained data from Flickr, a photo sharing site, and compared the quality and processing time of clustering between the proposed method and DBSCAN.

Keywords: density clustering, geosocial data, Bentley-Ottman algorithm, trajectory

1. はじめに

インターネットを介して共有されるビッグデータは，SNS と IoT という 2 つのブームを経て，爆発的に増加している．Twitter, Inc. [1] によると，2019 年 7 月から 9 月におけるアクティブユーザ数は 1 億 4,500 万人であると報告されている．また，Microsoft [2] 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．

さらに，近い将来にもたらされるであろう MaaS や自動運転を含めた，社会のデジタルトランスフォーメーションを考えると，ビッグデータが内包する地理情報を軸としたデータ分析技術の確立は急務であるといえる．

地理情報を有するソーシャルビッグデータの分析によって得られる知見の例として Sakaki ら [3] の研究がある．この研究では，地震などのイベント抽出を，地震計ではなく，

[1] 東京都立大学大学院システムデザイン研究科
Graduate School of Systems Design, Tokyo Metropolitan University, Hino, Tokyo 191–0065, Japan
[a)] ito-kotaro@ed.tmu.ac.jp
[b)] shohei@tmu.ac.jp

SNS ユーザの投稿を分析することによって行えることを示した．また Kan ら [4] は，タクシーの GPS データから交通渋滞の検出が可能であることを示した．このように，社会の様々な場面において，地理情報を有するソーシャルビッグデータの分析が寄与する余地がある．

本論文は，このような地理情報を有するソーシャルビッグデータ分析に主眼を置いた，革新的な密度クラスタリングアルゴリズムを提案する．ソーシャルビッグデータは地理空間上に離散的に存在するデータであり，多くの場合で，その分析は，広大な空間に分散している離散データから，高密度のクラスタを発見することが第 1 歩となる．

この高密度なクラスタとは，その領域内で多くのデータが存在することを意味している．たとえば，撮影場所がジオタグとして付与された写真のビッグデータにおいては，人々がたくさん写真を撮影している箇所であり，クラスタは人々の興味の地理的な分布を反映していると考えられる [5]．

空間上の高密度区域を発見するためのアルゴリズムには，しばしば Ester ら [6] の提案した DBSCAN（Density-Based Spatial Clustering of Applications with Noise）が用いられる．DBSCAN は近傍の点を用いて密度を計算し，その密度がユーザが与えた閾値以上ならば，クラスタを形成するアルゴリズムである．このアルゴリズムは，単に地理的なクラスタを発見するだけでなく，外れ値の除去も同時に行えることが特徴的であり，多くのノイズを含むソーシャルビッグデータの分析とは非常に親和性が高い．

しかしながら，DBSCAN 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る．この 2 つの値は，分析対象のデータの密度に関連したパラメータであり，高密度の区域を発見したいのにもかかわらず，あらかじめ密度を知っていなければ，最適なパラメータを発見できないという大きな制約がある．多くの SNS ユーザの集合知として作り出されるソーシャルビッグデータにおいて，データの粗密は事前に分からないことも多く，この制約は，分析の困難さの要因となっている．つまり，研究の多くの時間が DBSACN の最適なパラメータ探しに費やされることとなる．

また，DBSCAN は，入力データに点群をとるため，点の順序に意味があるデータを扱うときに，その意味を無視してしまう．具体的には「距離は近いが，移動軌跡がない 2 点」を同一クラスタに振り分けてしまうことがある．2 点間に移動軌跡がないことには何らかの理由があると考えられる．たとえば入力がジオタグのような緯度経度情報を含むデータの列である場合，軌跡のない 2 点間には，壁や川といった障害物の存在が考えられる．**図 1** に DBSCAN によるクラスタリングのイメージを，**図 2** に理想的なクラスタリングのイメージを示す．図 1 および図 2 において，

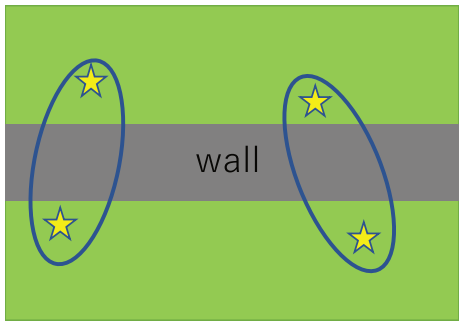

図 1　DBSCAN によるクラスタリングイメージ
Fig. 1　Clustering image with DBSCAN.

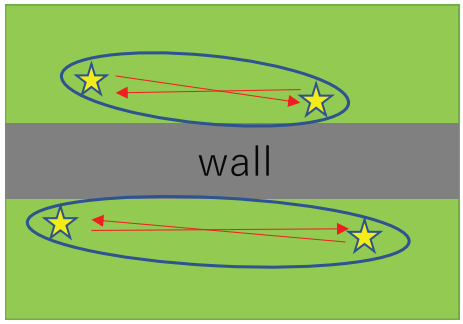

図 2　理想的なクラスタリングイメージ
Fig. 2　Ideal clustering image.

星印はデータがある地点，青の楕円はその内部のデータが同一クラスタに属することを表し，赤の矢印は移動軌跡を示す．このように，DBSCAN では移動軌跡がない 2 点でも同一クラスタに振り分けてしまうことがある．

Liu ら [7] や Hu ら [8] のような，ホットスポットの抽出を行う研究において DBSCAN を用いた場合，図 1 のように本来別のホットスポットとして抽出したい 2 地点を同じホットスポットとして抽出してしまう可能性がある．この結果は理想的な結果とはいい難く，障害物を越えずに移動ができるような地点を同一のホットスポットとして抽出するような結果を求められていると考える方が自然である．そのため，図 2 に示すような結果の方が，図 1 の結果よりも理想的であるといえる．つまり，人の移動のデータは，点群ではなく，線群であり，DBSCAN により作り出されたクラスタには，人の移動に関する情報は含まれていないため，理想的な結果を得ることが難しい場合がある．

これらの問題を解決するためのアルゴリズムとして，我々 [9] は EBSCAN（Entanglement-Based Spatial Clustering of Applications with Noise）という密度クラスタリングアルゴリズムのアイデアを提案した．これは移動軌跡の絡まりに着目することで，実際の地形を考慮に入れたアルゴリズムとなっている．

さらに DBSCAN ではデータの密度が高い場所を発見するために，密度を定義するためのパラメータを決定する必要があり，データの分布が不明な状態で適切に値を設定するのは困難である．EBSCAN では設定が必要なパラメー

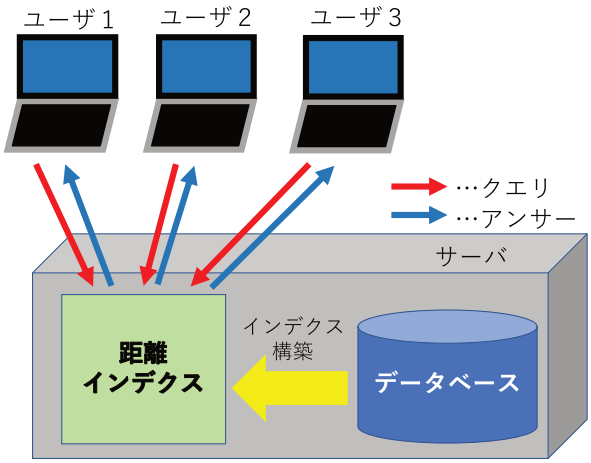

図 3　想定される使用環境
Fig. 3　Expected usage environment.

タは距離に関する 1 つの値 *toofar* のみとなっている．この *toofar* は，あるユーザの移動距離が *toofar* 以上であれば，それは地理的に違うクラスタへ移動したことを意味するパラメータであり，データの密度にかかわらず，直観的な設定が可能となり，利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．

もちろん，パラメータの数を半減したとしても，最適なパラメータの探索を行う必要はあるが，EBSCAN では，このパラメータ探索についても，DBSCAN に対して大きなアドバンテージを持っている．DBSCAN では，パラメータを変更すると，すべての点に関して再び密度の調査を行う必要がある．その結果，再計算に高いコストがかかってしまう．最適なパラメータ探索には，単位時間あたり計算量，すなわちスループットが重要であるが，DBSCAN では高いスループットを達成することは困難である．

また，来るべき社会のデジタルトランスフォーメーションにおいては，ソーシャルビッグデータ分析は，これまでのバッチ処理のような利用ではなく，**図 3** のような，クラウドとして多数のユーザが同時接続する環境が想定される．このような環境においては，同じビッグデータに対して，任意のパラメータを使った複数の処理の同時実行が前提となり，アルゴリズムとしても，クエリに対する処理時間を可能な限り短縮し，スループットの向上を目指すことは必須の課題である．

本論文では，移動軌跡の交わりに着目したインデクス法と，そのインデクスに対する，密度クラスタリングを行うアルゴリズムを提案する．提案するインデクスは異なる粒度の密度クラスタリングに対しても再構築する必要がなく，また 1 度インデクスを構築すれば，クラスタリングは非常に高速に実行できる特徴を持っている．また構成されるクラスタには，人の移動が考慮され，地理的かつ社会的な「まとまり」を抽出することができる．

EBSCAN では，軌跡の交点を発見するために，著名な交点発見アルゴリズム Bentley-Ottman を用いている．Bentley-Ottman は主にコンピュータグラフィックなどに用いられるアルゴリズムであり，我々はこれをクラスタリングアルゴリズムとして利用している．Bentley-Ottman は計算誤差に弱いアルゴリズムとして知られており，本研究では実験を通じて，精度と処理速度の検討を行った．また，本論文では提案する EBSCAN の有用性を示すために，実データと人工データを用いた実験を行い，DBSCAN との比較においてスループットとクラスタの質の評価を行った．

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，EBSCAN の詳細な構成を述べる．4 章では EBSCAN を実現するうえで問題点となりうる，計算精度に依存する問題点とその解決方針を述べる．5 章では処理時間に関する実験および考察を述べ，6 章ではクラスタリング結果に関する実験および考察を述べる．7 章で本研究のまとめと今後の課題を述べる．

2. 関連研究

2.1 密度ベースのクラスタリング手法

密度ベースのクラスタリングに関する研究の代表的な例として，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 [10] 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，関数を用いて距離が離れている点に対して低い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 [11] 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あわ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．しかし，この研究ではノイズの多いデータに手法を適応させるためには，パラメータの数を増加させる必要があるため，パラメータの設定に関する問題が完全に解決されているとはいい難い．

密度ベースのクラスタリングアルゴリズムにおいては，パラメータの設定に関する研究もさかんに行われている．

Ankerst ら [12] は，データセット中の異なる密度のクラスタを発見するためのパラメータ分析の手法として，OPTICS を提案した．これは，データセットに対してオブジェクト

が処理される順序と番号を振りあてるための情報をあわせ持ったインデクスを構築しておくことで，密度ベースのクラスタリング構造を表すのに有効なことを示している．本研究でもこの論文と同様にインデクスを構築し，後の密度クラスタリングで活用を行うが，OPTICS はインデクスをパラメータ分析に活用することに主眼を置いているのに対し，本研究ではインデクスをクラスタリングの高速化に活用することに主眼を置いた．

Esmaelnejad ら [13] は DBSCAN のパラメータである ϵ を適切に決定するために，ノイズとなるオブジェクトの点の数やノイズ比を用いる方法を提案した．しかし，この研究では ϵ の決定においてもう 1 つのパラメータである *MinPts* に依存しているため，*MinPts* の適切な値を求める必要があることや，ノイズ比が分からない場合にはこの手法を適用できないなどの問題点がある．

Karami ら [14] は DBSCAN の 2 つのパラメータを適切に決定するために差分進化アルゴリズムを導入する方法を提案した．この手法では，パラメータの最適化のためにループ処理を用いるため，最適なパラメータを求めるまでに時間がかかることが課題として残っている．

2.2　ジオソーシャルデータを対象としたクラスタリング手法

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 [15] 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の発見を可能とした．本論文ではこの論文と同様に，高品質な（地形情報を反映しているなど，意味のある）クラスタの抽出を行うことも目的の 1 つである．

Kisilevich ら [16] 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．この研究では近傍の密度を計算する際に，写真の点数そのものではなく，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，コア点に関する密度の変化率を考えることで急激に密度が変化した場所の検出を可能とした点が DBSCAN と異なる．本研究では，この論文と同様にジオタグ付き写真のデータに対して高密度な領域を発見することを目的としている．

2.3　本研究の新規性および有効性

このように，密度ベースのクラスタリングアルゴリズムは多数存在し，ジオソーシャルデータの分析に関してもそのようなアルゴリズムを用いることがしばしばある．本研究ではそのようなジオソーシャルデータの分析において，パラメータの数を削減し，クエリに対する時間を短縮したうえで，クラスタリング結果を改善するといった 3 点の問題を改善するアルゴリズムを提案する．

まず，パラメータ数の削減に関しては，移動軌跡を用いることで達成を目指した．提案手法は 1 章で述べたように，移動軌跡の交点に着目する．データが集まっている領域は，移動軌跡が交点を持つ（絡まる）ことが多い．したがって，複数の移動軌跡が交点を持つ場所に着目することで，ある程度データが集まっている領域を発見するという目標を達成できる．そのため，ユーザは移動軌跡の端点（データ）間の距離を指定するという直感的な操作のみで，高密度な領域を発見可能となる．

そのため，本研究で提案するアルゴリズムの入力データは軌跡群に限定されるが，前述した DBSCAN や P-DBSCAN を利用してジオソーシャルデータの分析する研究に比べ新たなデータは必要としない．なぜならば，ジオソーシャルデータは本質的には軌跡である．たとえばジオタグ付きツイートのビッグデータはユーザ単位で時系列に並べると，すなわちそれはユーザ軌跡となる．従来のジオソーシャルデータ分析では密度クラスタリングとして活用していなかった移動軌跡の情報を入力とすることで，従来よりもパラメータレスな密度クラスタリングを行うことを目的とする．

また，移動軌跡を用いるというアイデアは，クラスタリング結果の改善にも深く関係している．「移動」という行動の性質上，壁や川などの横断が困難な障害物を越える軌跡は少ないと考えられる．したがって，移動軌跡を用いることで，図 2 のように地形を考慮に入れたクラスタリングが可能となる．

最後に，クエリに対する時間短縮については，密度クラスタリングを行う過程で，距離に関するインデクスを作成することにより達成した．このインデクスは移動軌跡の交わりによって構成され，交点を持つ移動軌跡の端点間の距離情報を保持している．そのため 1 度構築を行えば，データの更新がない限りは再構築が不要である．したがって，異なるパラメータを持つクエリに対しては，データ間の距離を再計算する必要がなくなり，高速な処理が可能となる．移動軌跡の交わりを求めるためには Bentley ら [17] の Bentley-Ottman アルゴリズムを用いた．

ここで，Bentley-Ottman アルゴリズムに関する研究を示す．Bentley-Ottman アルゴリズムは掃引線を用いて走査することで，入力線分の全交点を発見するアルゴリズムである．Bartuschka ら [18] は掃引線の形や，線分を管理するためのシークエンスの内容を改良したアルゴリズムを提案した．これにより，従来のアルゴリズムでは処理が正常に行われなかった場合でも，交点が発見可能であることを示した．本研究でもこの論文と同様に x，y の 2 段階で優先度をつけることで，掃引線の方内を工夫し，交点を探

索した．

Boissonnat ら [19] は代数の次数という定義を用いることで，Bentley-Ottman アルゴリズムと同様の計算量でありながら，必要とする入力精度が小さくて済むアルゴリズムを提案した．しかし，この研究では実装に関する言及が少なく，実際の処理時間に関するデータは示されていない．また，計算精度の問題に関する研究として，Benouamer ら [20] は遅延ライブラリを用いることで，入力精度が高くなくても交点を発見できることを示した．

Bentley-Ottman アルゴリズムは，計算誤差により実装が困難であることが知られている．そのため，本研究ではこれらの論文をふまえたうえで実装を行い，その際に問題となるような場合を紹介する．また，その問題の解決方針を示すことで，以前の論文 [9] と比較してアルゴリズムの実現性が改善されたことを示す．

3. EBSCAN の構成

図 4 に EBSCAN の概要を載せる．EBSCAN の手順は大きく分けて 3 種類のタスクから構成される．

まず，軌跡計算タスクを行う．軌跡計算タスクはデータ取得ステップと移動軌跡の計算ステップから構成される．データ取得ステップでは API などを用いてインターネット上の緯度経度情報付きの写真に関するデータを取得する．データ取得後，移動軌跡の計算ステップで，そのデータを用いてユーザの移動軌跡を計算する．ここで，写真の撮影位置は計算した移動軌跡の端点として扱われる．

次に，移動軌跡の緯度経度情報を地理座標系からデカルト座標系に変換し，交点探索タスクに移る．交点探索タスクでは計算された移動軌跡を入力とし，それらのデータの交点を出力する．

最後に，発見された交点の座標を地理座標系に戻し，クラスタリングタスクを実行する．クラスタリングタスクは端点間の距離計算ステップとクラスタリングステップから構成される．端点間の距離計算ステップでは，交点を有する線分の組合せに対して端点間の距離を計算する．クラスタリングステップではその距離が事前に設定した閾値未満である場合，端点（写真）の密度クラスタリングを行う．

また，図 2 に示すように EBSCAN は距離インデクス構築フェイズとクエリ応答フェイズに分類できる．データの変更がない限り距離インデクス構築フェイズは 1 度のみ実行し，クエリ応答フェイズはクエリごとに実行を行う．

3.1 データの取得

本研究では Flickr が提供している 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．取得するデータの内容は，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，および撮影日時の情報である．**図 5** に Flickr で取得する情報の例を示す．また，**図 6** に Flickr で取得する画像の例を示す．

本研究では緯度経度情報を指定し，その領域内の画像データの取得を行った．

3.2 移動軌跡の計算

このステップでは，取得したデータを用いて移動軌跡の計算を行う．まず，3.1 節で説明したように，Flickr API を用いて，ユニークな画像を集めた．

次に，ユーザ ID を用いて全画像データを撮影したユーザごとにまとめる．さらに，各ユーザが撮影した画像を活動単位でまとめる．ここで，各ユーザにおいて撮影した画像を撮影時刻順で並べ替え，連続する 2 つの画像の撮影時刻の差が一定の値以内だった場合，その画像は同じ活動に属すると見なす．本研究で用いたデータは，ユーザによって撮影時刻の間隔にばらつきがあった．そのため，軌跡デー

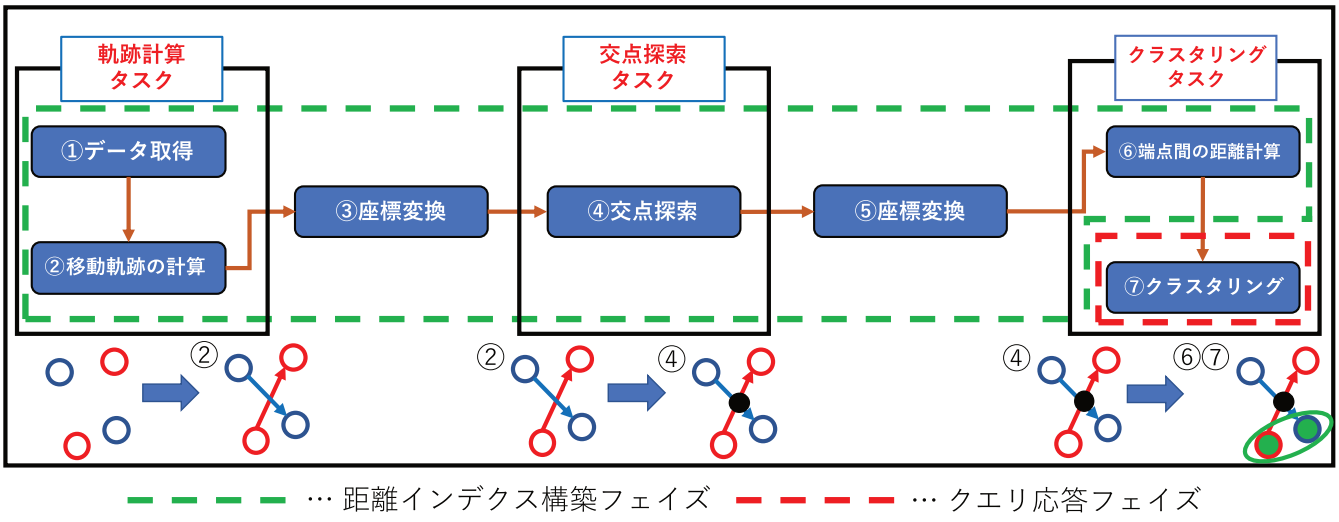

図 4　EBSCAN の概要図
Fig. 4　Overview of EBSCAN.

photoID	ユーザID	緯度	経度	撮影日時
484218626	503266@N05	35.299047	139.455272	2017/1/21/ 11:56:45
484218624	503266@N05	35.299063	139.475271	2017/1/21/ 11:39:58
482800816	61758704@N06	35.299701	139.478572	2019/5/11/ 15:32:31
482714084	61758704@N06	35.299563	139.477697	2019/5/11/ 14:44:51
482566138	61758704@N06	35.299158	139.475019	2019/5/11/ 13:01:47
481866070	979028@N02	35.299858	139.480636	2019/7/3/ 16:58:25

図 5　Flickr で取得する情報例
Fig. 5　Information example to get with Flickr.

図 6　Flickr で取得する画像例[*1]
Fig. 6　Image example to get with Flickr.

タの長さをある程度揃えるために，撮影時刻の間隔が 12 時間以上離れた場合は別の活動と見なしたが，活動の単位については本アルゴリズムを利用するアプリケーションによって決定してよい．画像を活動単位でまとめた後，画像が 1 枚しかない活動に関しては軌跡を生成できないため，その画像は消去する．

そして，すべての活動の画像をまとめ，撮影時刻順にチェックしていく．ある活動に属する 2 枚目以降の画像が出現した際に，その画像とその画像の直前に出てきた同一活動に属する画像 1 枚とで軌跡を生成し，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することが可能となる．

最後に，軌跡の 2 端点が同じ座標の場合，その軌跡を消去する．Flickr から取得するデータのうち，一部のユーザではアップロードしているすべ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 1 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生成された時期が後の軌跡を除去する．

[*1] © “江ノ島 - Enoshima DSC00791” by wellflat. is licensed under CC BY 2.0 (2014).
https://www.flickr.com/photos/95962563@N02/15662060658/

3.3　座標変換

本研究では交点探索フェイズの前後で座標変換を行う．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 度未満と小さく，小数点第 6 位までのデータにて線分の交差判定を行う場合，除算による浮動小数点誤差由来の計算誤差が発生した際に，3.4.1 項にて後述する Bentley-Ottman アルゴリズムのイベントの発生順序が逆転し，イベントの発生順序が崩れてしまうことがある．そういったイベントの発生順序の逆転が起きる回数を少なくするため，以下の式で地理座標系からデカルト座標系への変換を行った．lon，lat はそれぞれ画像を撮影した地点の経度と緯度，$minlon$，$minlat$ はデータを取得した領域の最小の経度と緯度である．

$$x = (lon - minlon) * 10^9 \quad (1)$$

$$y = (lat - minlat) * 10^9 \quad (2)$$

この状態で 3.4 節の交点探索を行い，その後逆変換の式を用いて地理座標系へ戻した．これにより，クラスタリングタスクでは地理座標系の緯度経度情報を用いて計算を行う．

3.4　交点探索

この節では，交点探索タスクで行う，交点探索について述べる．構成としては，まずもととなるアルゴリズムである Bentley-Ottman アルゴリズムについて述べる．次に，Bentley-Ottman アルゴリズム内でツリーを管理するために用いたデータ構造である赤黒木について述べる．

3.4.1　Bentley-Ottman アルゴリズム

Bentley-Ottman アルゴリズムは Bentley らが提案した，入力線分に関するすべての交点を求めることが可能なアルゴリズムである．このアルゴリズムは走査法に基づいており，y 軸に平行な掃引線が x 軸上を一定方向に移動していくことで進行する．掃引線の移動により，線分の端点もしくは交点と掃引線が重なった場合，その点の種類に応じてイベントが発生する．

イベントの種類に応じて，掃引線が位置している x 値上に存在する各線分の y 値の大小関係を比較し，線分の上下の情報，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直上の線分や直下の線分との交点の有無のみを計算することが可能となる．したがって，遠く離れた線分との交点を探索せず，効率良く交点探索が行える．なお，イベントは以下のとおりである．

(1) 点が線分 $l1$ の左端点
(2) 点が線分 $l1$ の右端点
(3) 点が線分 $l1$ と線分 $l2$ の交点

また，実装上の問題により，本研究では従来の 3 種類のイベントに加えて，以下のイベントを追加した．これによ

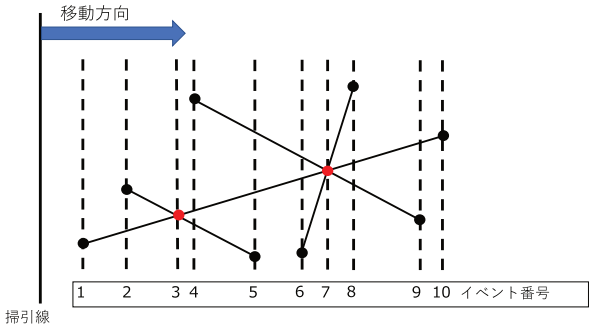

図 7　Bentley-Ottman アルゴリズム
Fig. 7　Bentley-Ottman algorithm.

り，複数の線分が 1 点で交わる場合を考慮した．

(4) 点が 3 線分以上の交点

図 7 は Bentley-Ottman アルゴリズムの動きの説明図である．この図において，Bentley-Ottman アルゴリズム内でイベントが発生する位置には点線を引き，対応する位置にイベント番号を記した．

Bentley-Ottman アルゴリズムに最初に入力される情報は線分の情報のみ，つまり端点のみなので，初期状態として左端点のイベント（図 7 におけるイベント 1，2，4，6）と右端点のイベント（図 7 におけるイベント 5，8，9，10）が与えられ，最も左に存在するイベントよりも左側に掃引線が配置される．その後，掃引線を右側に動かしていき，それぞれのイベントで，そのイベントに関連する線分と，その上下の線分との交点が存在するかを確認し，2 線分の交点（図 7 におけるイベント 3）や 3 線分以上の交点（図 7 におけるイベント 7）を発見していく．ここで発見した交点はその座標情報および，その交点を構成する 2 線分の線分情報を持つ．

掃引線を移動させていき，すべての線分の端点および交点のイベントを処理が終了したら，このアルゴリズムは終了となる．なお，イベントは座標の位置で発生する順序が決定されるため，イベントを管理する際には優先度付きキューを実装する必要がある．したがって，イベントの管理にはヒープツリーのデータ構造を用いて，優先度付きキューを実装した．

3.4.2　赤黒木

本研究では Bentley-Ottman アルゴリズム内における線分の位相を管理する際に赤黒木を用いた．赤黒木は平衡二分木の一種であり，ノードに赤と黒の色を付けたうえで，複数の制約条件を課すことで「根から葉までの道の中で最長のものは，根から葉までの道の中の最短のものの長さの 2 倍以下である」という性質を保証している．赤黒木は探索，挿入，削除などの操作の最悪時間計算量 O を $O(\log N)$ で行えるとしており，実用的なデータ構造である．ここで，N はツリーの要素数を表している．

また，各ノードにはキーおよびバリューが存在し，本研究ではキーに y 値，バリューに線分の ID 情報を格納した．ここで，キーの値は重複が許されていないことから，交点イベントが発生した場合，同じノードに複数の線分を格納するように実装を行った．

3.5　端点間の距離計算

クラスタリングタスクの初めに，交点探索タスクで発見した交点に関する線分の端点間の距離計算を行う．交点を構成する軌跡をリストから取り出し，その軌跡の端点の情報を調べる．その端点の 4 つの組合せに関して，2 点間の距離を計算する．本研究では緯度によらずに正確な距離を算出できる Hubeny の公式 [21] を用いて緯度経度から距離計算を行った．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．

計算した結果は順次リストに追加していき，最終的に完成したリストを距離インデクスと定義する．

3.6　クラスタリング

このステップではまず 2 点間の距離に関するパラメータ *toofar* を設定する．このパラメータはクラスタリングにおける閾値となる．

次に，すべての端点（画像）の初期状態として，所属状態を未所属に設定する．続いて 3.5 節で作成した距離インデックスを参照することで 2 端点を取得する．これらの 2 端点間の距離が *toofar* 以下の場合，それらの端点は密度クラスタリングの対象となる．ここで，クラスタリングの対象となった点がいずれかのクラスタに属しているかどうかを調べ，所属状況に応じた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 2 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．

クラスタリングの方法は端点の所属状況に応じて以下のように分岐する．

(1) どちらの端点もクラスタに属していない場合

新規のクラスタを作り，そのクラスタに 2 端点を追加する．所属済み端点リストにその 2 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す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 2 端点での距離を比較し，現在の 2 端点での距離の方が短ければ，クラスタの更新を行う．

(3-a) 片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．

これらのルールに基づき，距離インデクスにある端点の組合せをすべて処理するまでクラスタリングを行う．その後，要素の移動によりクラスタ内の要素数が 1 以下とな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，要素数が 1 になったクラスタの要素以外に，距離インデクスに出現しない端点や，他の軌跡との交点を持つが，その軌跡の 2 端点間の距離が *toofar* よりも大きいような組合せしか持たない端点は外れ値として扱い，どのクラスタにも属さない点となる．したがって，最終的には入力された写真の座標（線分の端点）はいずれかのクラスタに属する要素もしくは外れ値としてクラスタリングされる．

4. 計算精度による誤差の問題

本研究では，交点探索フェイズで，Bentley-Ottman アルゴリズムを用いる．このアルゴリズムは線分の交差判定を行う際に，除法を用いて計算するため，精度の高い型を用いて演算を行う必要がある．

しかし，この場合，通常の精度の型での演算と比べて処理時間がかかるという欠点がある．仮に通常の精度の型での演算を工夫せずに行った場合，交差判定が正常に行われず，線分の交差イベントの欠落や，線分の位相管理が正常に行われなくなるという問題が発生する．この問題が発生した場合，Benley-Ottman アルゴリズムは逐次的なアルゴリズムであるためアルゴリズム全体が正常に動作しなくなるといった恐れがある．

そこで，本研究では通常の精度の型で演算を行う際に出現しうる問題点を列挙し，それに対する解決策を考えることで，従来の精度の高い型を用いた演算よりも高速に線分の交点を発見可能とし，アルゴリズム全体の処理時間を短縮した．

まず，精度の低い型で演算を行う場合の変更点を説明する．線分の交差判定を行う際，交点の座標を計算し，それによって 2 線分が交差しているかどうかを判定する．

本研究では矩形の 4 つの頂点を決定し，計算された交点の座標がその矩形の範囲内に存在するときに，線分が交差していると判定した．頂点の決定手順としてはまず，線分の端点を通り，y 軸に対する垂線を引く．ここで，上にある垂線から順にそれぞれ，垂線 A，B，C，D とする．次に，2 線分の x 値が大きい方の左端点を通り x 軸に対する垂線アと，2 線分の x 値が小さい方の右端点を通り x 軸に対する垂線イを引く．これらをもとに，以下の 4 点を決定する．

(1) 垂線アと垂線 B の交点
(2) 垂線アと垂線 C の交点
(3) 垂線イと垂線 B の交点
(4) 垂線イと垂線 C の交点

図 8 に交差判定の領域決定例を示す．

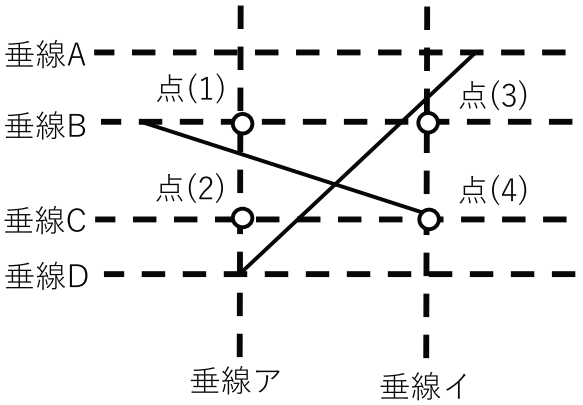

図 8　交差判定の領域決定
Fig. 8　Area determination for intersection judgment.

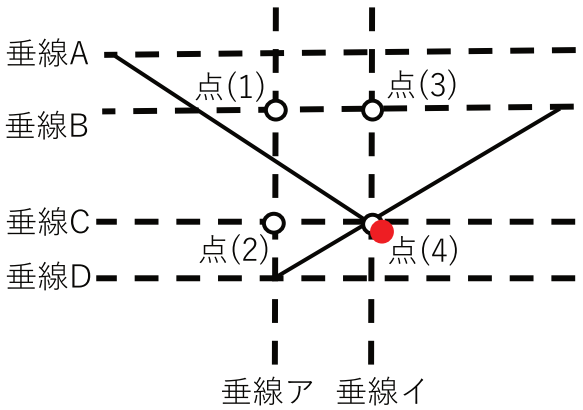

図 9　交差していないと判定される例
Fig. 9　Example where it is judged that they do not intersect.

しかし，精度の低い型を用いて演算を行った場合，浮動小数点由来の計算誤差によって交点の座標が本来の位置とずれてしまい，実際は線分が交差しているのに，交差していないと判定されることがある．**図 9** にその例を示す．

図 9 は，浮動小数点由来の計算誤差により，交点が赤い丸の位置にある，つまり，本来の位置よりも右下に交点の座標が存在すると計算された例である．この場合，点 (1) から点 (4) までの矩形内の領域に交点は存在しないため，実際に線分が交差していても，この 2 線分は交差しないという判定になる．

この問題を解決するために，本研究では tolerance とい

う誤差区間に関する値を導入した．tolerance は想定される計算誤差の範囲を示す．この値を用いて矩形の領域を広げることで，計算誤差によって実際の位置から移動した交点を，矩形の領域内に収めることができる．

次に，tolerance を用いることによって生じる問題点について述べる．tolerance を用いることで生じる問題点は大きく分けて 2 つ存在する．

考えられる問題点の 1 つ目として，tolerance を用いることで矩形が拡張されるため，交点が拡張された領域に存在すると計算されることがある．その際，線分が実際には交差していない場合でも，それらの線分は交差していると判定される．この問題が起きた場合，交点の位置によってはアルゴリズムにエラーが生じる．交点の x 座標が (1) の左端点よりも小さい場合は，掃引線が一時的に逆行するだけであり，致命的なエラーにはならない．しかし，交点の x 座標が (2) の右端点よりも大きい場合は，線分の終了イベントの後に交差イベントが存在するため，その交差イベントにおいて，線分を管理しているツリー内に該当する線分は消去されている．そのため，線分を管理するツリー内での探索を行っても，該当する線分が発見できずにエラーが発生し，アルゴリズムがその地点で停止してしまう．

この問題に関しては，2 つの解決法が考えられる．まず，1 つは計算された交点の位置を修正する方法である．計算された交点の位置が線分の端点より外側だった場合，線分の端点上に位置を修正する．こうすることで，終了イベントと交差イベントが同座標上に存在するため，イベントの種類に優先度を設定することで交差イベントが終了イベントよりも先に発生し，ツリー内に線分がある状況で終了イベントが発生する．問題点としては，実際に交差していない線分が交差すると見なされ，交点の個数が実際よりも多く出力されることである．しかし，「計算誤差によって交点が存在する」といった状況になるのはそもそも 2 線分が非常に近い場合であり，入力データのわずかな誤差によって，交点が存在していてもおかしくないケースが多い．そのため，大元のアイデアである「移動軌跡の絡まりに着目することで，実際の移動を考慮に入れる」という考え方から大きく逸脱するものではない．

次に考えられるのが，エラーが発生した地点から逆方向に掃引線を動かす方法である．この方法では，エラーが発生した時点で 1 度アルゴリズムを停止させ，ツリーの情報をリセットする．そして停止した地点から逆方向に掃引線を動かしていき，停止した地点のイベントに関連する線分が関わる別のイベントが出現する地点まで計算を行う．その後停止した地点に戻り，再び順方向に掃引線を動かす．この方法の狙いは，エラーが発生した地点から逆方向に掃引線を動かすことで，エラーが発生したイベントとエラーの原因となったイベント間の領域に存在するイベントを正常に動作させることである．問題点としては，逆方向に計算を行うことやツリーの情報を再構築するため，処理時間が余計にかかるということがあげられる．

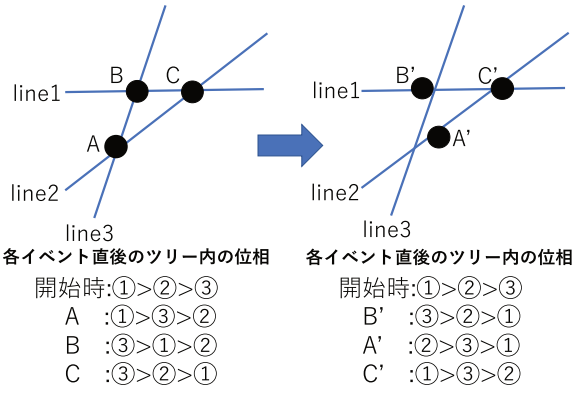

図 10　計算誤差によるイベント順序変更例

Fig. 10　Example of changing the event order due to calculation error.

続いて，tolerance を導入することによって生じるもう 1 つの問題点について述べる．ある交点の近傍に交点が存在する場合，計算誤差によってイベントの順序が変更されることがある．**図 10** にその例を示す．図 10 は line1，2，3 の線分と，それらの線分によって作成される交点 A，B，C，およびその交点（2 線分の交差イベント）直後での線分の位相（線分の上下関係）を表すものである．なお，図 10 内における位相の丸数字の中の番号はそれぞれの線分（line）の番号と対応している．

図 10 では交点 A，B，C が計算誤差によって A’，B’，C’ の位置にずれた結果，イベントの発生順序が入れ替わってしまっている．その結果，各イベントでの位相交換も発生順序が変わり，各イベントが発生した直後の線分の正しい位相情報を保てなくなっている．この状況でアルゴリズムを進めると，各イベントで交点が正常に追加されなくなるという問題が発生する．

この問題に関して，交点を計算した時点で感知するのは難しい．そのため，この問題の解決法として交点イベント内において感知を行う方法が考えられる．具体的には，ツリーから取得した線分の上下関係と線分の傾きを用いる．交点イベントの開始時点では，その交点に関わる 2 線分のうち，位相が下の線分の方は位相が上の線分よりも傾きが大きいはずである．しかし，図 10 を見て分かるように，交差イベントの計算順序が入れ替わると，位相が下の線分の方が傾きが小さくなる．したがって，この時点で問題が発生していることが分かる．問題の発生を検知した後は，先述したもう 1 つの問題のときと同様に，1 度逆方向に掃引線を動かし，その後再び該当イベントから順方向に掃引線を動かすことで，正常な位相情報を保ちつつ，交点を探索できる．

このような工夫をすることで，通常の精度の型を用いた計算でも問題なく交点を探索できると考えられる．

5. 処理時間に関する実験

5.1 データセット

まず，同一のデータに対して EBSCAN と DBSCAN を適用し，その処理時間の比較を行う．DBSCAN の計算量は $O(n \log n)$ であり，提案手法で用いている Bentley-Ottman アルゴリズムの計算量が $O((n+k)\log n)$ である．ここで，n はデータ数，k は交点の数である．したがって，人工データの場合，一方が有利になるデータを容易に作り出せるという問題があるため，ここでは実験対象として実データを選択し，実験を行った．

実データは Flickr API を用いて 3 種類のデータを取得した．データの種類はデータ 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 度で撮影され，2018 年 5 月 9 日から 2019 年 10 月 22 日の間にアップロードされたデータ 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 度で撮影され，2019 年 12 月 13 日から 2020 年 1 月 21 日の間にアップロードされたデータ 2,500 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 度で撮影され，2016 年 1 月 2 日から 2020 年 1 月 15 日の間にアップロードされたデータ 6,551 件である．

5.2 実験結果

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．高精度の Bentey-Ottman アルゴリズムの実装は Python のライブラリである shapely [*1] を用いた．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．

次に，EBSCAN の各ステップごとの処理時間を**表 2** に示す．ここで，交点探索は表 1 内における最小の処理時間を記載し，クラスタリングステップは $toofar = 35\,\mathrm{m}$ に設定した際の処理時間を記載した．

続いて，R-tree を用いて実装した DBSCAN について，各段階ごとの処理時間を**表 3** に示す．ここで，クラスタリングタスクは $\epsilon = 12.5\,\mathrm{m}$，$MinPts = 5$ に設定した際の処理時間を記載した．

最後に，EBSCAN と DBSCAN のクラスタリングフェイズにかかる処理時間の比較を**表 4** に示す．

5.3 考察

まず，交点探索に関する時間について考える．Bentley-Ottman アルゴリズムを用いて実装したもののうち，高精度の浮動小数点の型を用いて計算を行ったものは，通常の精度の浮動小数点を行ったものに比べて 4 倍前後の時間がかかることが分かった．

[*1] https://github.com/Toblerity/Shapely

表 1　交点探索の処理時間
Table 1　Processing time at Intersection search.

	データ 1 [秒]	データ 2 [秒]	データ 3 [秒]
総あたり	0.174663	1.058569	12.77074
B&O（高精度）	0.651823	1.085119	12.17605
B&O（通常精度）	0.151077	0.284516	3.794537

表 2　EBSCAN の各ステップにおける処理時間
Table 2　Processing time at each step of EBSCAN.

	データ 1 [秒]	データ 2 [秒]	データ 3 [秒]
移動軌跡の作成	0.005207	0.026242	0.142616
座標変換 1	0.002084	0.008573	0.009944
交点探索	0.151077	0.284516	3.794537
座標変換 2	0.003125	0.010415	0.090732
距離計算	0.018003	0.033993	0.501111
クラスタリング	0.003032	0.003993	0.058776

表 3　DBSCAN の各段階における処理時間
Table 3　Processing time at each phase of DBSCAN.

	データ 1 [秒]	データ 2 [秒]	データ 3 [秒]
R-tree 構築	0.021243	0.046884	0.221413
クラスタリング	0.093730	0.127965	0.719465

表 4　クラスタリングの処理時間の比較
Table 4　Comparison of clustering processing time.

	データ 1 [秒]	データ 2 [秒]	データ 3 [秒]
EBSCAN	0.003032	0.003993	0.058776
DBSCAN	0.093730	0.127965	0.719465

一方，総あたりで計算したものは，データ 1 では処理時間が 2 番目に小さかったが，データ 3 では高精度での Bentley-Ottman よりも大きい処理時間となった．これは，計算対象となる線分の数が増加したことが原因であると考えられる．この場合，高精度での Bentley-Ottman とは同じくらいの処理速度であるが，通常精度での Bentley-Ottman よりも大きな処理時間となっている．これらのことより，ツリー構造を用いた通常精度の Bentley-Ottman アルゴリズムは交点探索において有用であることが示された．

次に，全体での処理時間について考える．EBSCAN と DBSCAN を比較すると，EBSCAN では交点探索ステップに時間がかかるため，全体の処理時間の合計も EBSCAN の方が大きな値となっている．しかし，クラスタリングステップのみを比較すると，EBSCAN の方が DBSCAN よりも小さな値となっている．これは，パラメータを変更して再クラスタリングをする場合，提案手法の方が低コストで計算ができるということを示している．

図 11 はインデクス構築とクラスタリングにかかる処理時間のグラフである．ここで，EBSCAN のインデクス構築は図 4 のインデクス構築フェイズにかかる処理時間の総和である．この図においてもとのデータが変更されない場

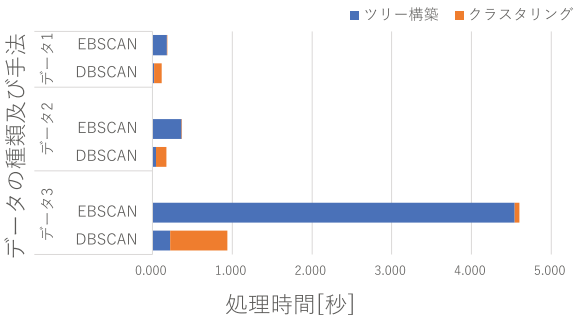

図 11　各データに対する処理時間の比較
Fig. 11　Comparison of processing time for each data.

表 5　1 分間あたりのスループット
Table 5　Throughput per minute.

	データ 1 [回]	データ 2 [回]	データ 3 [回]
EBSCAN	14,735	13,939	1,343
DBSCAN	633	477	80

合，インデクス構築フェイズは 1 度計算すれば良いが，クラスタリングフェイズはパラメータを変更してクラスタリングを実行するたびに計算する必要がある．

表 5 は入力パラメータをランダムに設定したクエリに対する 1 分間あたりのスループットを測定した結果を示している．表 5 を見ると，同一時間で EBSCAN は DBSCAN よりも多くのクエリを処理できることが分かる．したがって，図 3 のような環境下，つまりユーザが同時接続を行って利用する場合，重要となるのはツリー構築部分にかかる処理時間よりもクエリに対するクラスタリング結果を出力する時間であるといえる．したがって，データの更新頻度よりもクエリの処理回数の方が多いような状況での運用を考えた場合，DBSCAN よりも EBSCAN の方が優れているといえる．

6.　クラスタリング結果に関する実験

6.1　データセット

続いて，クラスタリング結果に関する実験を行う．ここでは，5 章で用いた実データを対象とした実験を対象とした定性的な実験と，人工的に作成したデータを対象とした定量的な実験の 2 つを行った．**図 12** に作成した人工データを示す．ここで作成したデータには 7 種類の正解ラベルが貼られており，図 12 中において同じ正解ラベルが貼られているデータは同色のマーカによって示されている．また，図 12 中の青の直線は移動軌跡を示している．このデータに対し，EBSCAN と DBSCAN を適用し，クラスタリング結果の精度を測定する．

6.2　定性的評価

6.2.1　実験

5.1 節で取得したデータ 1 に対し，EBSCAN と DBSCAN

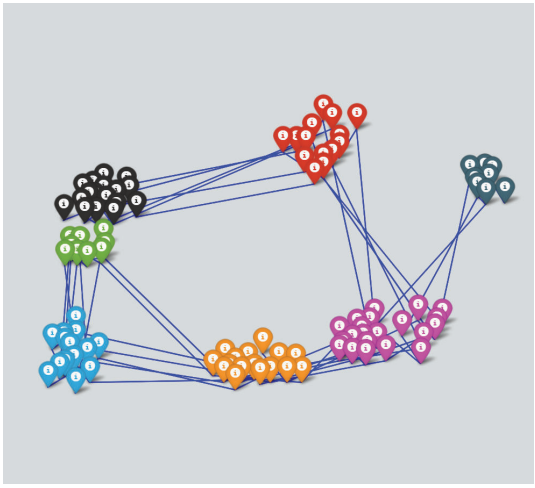

図 12　作成した人工データ
Fig. 12　Created artificial data.

図 13　データ 1 に対する EBSCAN（$toofar = 35\,\mathrm{m}$）のクラスタリング結果
Fig. 13　Clustering result of EBSCAN ($toofar = 35\,\mathrm{m}$) for data1.

の両手法を用いてクラスタリングを行った．各手法で出力されたクラスタの要素（写真）をクラスタごとに色分けして，実際の地図上に表示した結果を示す．**図 13** が EBSCAN によるクラスタをマッピングしたものである．

一方，**図 14** および**図 15** はそれぞれ異なるパラメータ設定をしたときの DBSCAN による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，同じクラスタに属するデータ（写真）は同一のマーカによってマッピングを行った．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，マーカによるマッピングは行わなかった．各図においてパラメータ設定は図 13 が $toofar = 35\,\mathrm{m}$，図 14 が $\epsilon = 12.5\,\mathrm{m}$，$MinPts = 5$，図 15 が $\epsilon = 17.5\,\mathrm{m}$，$MinPts = 5$ とした．

6.2.2　考察

前項で行った実験において，図 13 から図 15 内の同一領域を赤の楕円と緑の楕円で囲んだ．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

図 14　データ 1 に対する DBSCAN（$\epsilon = 12.5\,\mathrm{m}$, $MinPts = 5$）のクラスタリング結果

Fig. 14　Clustering result of DBSCAN ($\epsilon = 12.5\,\mathrm{m}$, $MinPts = 5$) for data1.

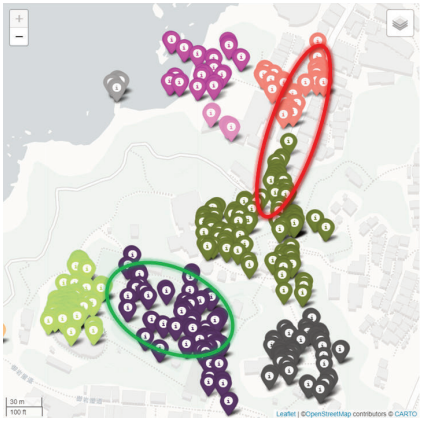

図 15　データ 1 に対する DBSCAN（$\epsilon = 17.5\,\mathrm{m}$, $MinPts = 5$）のクラスタリング結果

Fig. 15　Clustering result of DBSCAN ($\epsilon = 17.5\,\mathrm{m}$, $MinPts = 5$) for data1.

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 13 より，EBSCAN では赤の楕円内部の多くを青色のマーカ 1 種類が占めていることが分かる．これに加えて，青色のマーカで赤色の楕円から外にはみ出しているものは少数であることが確認できる．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が少ないことが分かる．以上のことから，EBSCAN では各領域を 1 つのクラスタとして抽出することに成功したといえる．

続いて，DBSCAN を用いた場合のクラスタについて確認する．図 14 では，どちらの楕円においても，楕円内を占めている色のマーカで楕円の外側に出ているものは少ない．しかし，赤の楕円内部には主に 3 種類のマーカが存在しており，緑の楕円内部では主に 2 種類の色のマーカが存在しているため，それぞれの領域を 1 つのクラスタとして抽出できているとはいい難い．

また，図 15 を見ると，緑の楕円内部は紫色のマーカが広域を占め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．そのため，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いえる．しかし，赤の楕円に関しては内部に主に 2 種類のマーカが存在しており，1 つのクラスタとして抽出できていない．加えて，赤の楕円内部の南側を占めるマーカが楕円の外部へ大きくはみ出し，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分かる．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いえる．

これらの違いについて考察する．赤の楕円が示す弁天財仲見世通りに関しては，通りの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中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 2 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大きくなる．そのため，異なる観光スポットを跨ぐような移動軌跡に関する写真の組合せは撮影地点間の距離が *toofar* よりも大きな値となりやすく，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のだと考えられる．その結果，DBSCAN では近傍点数が足りずに別のクラスタとして出力されるようなクラスタが，EBSCAN では結合し，出力されたのではないかと考えられる．

このように，本研究の実験では観光スポットという括りで実際に人が移動したと考えられる経路に沿ったクラスタを発見することに成功した．このことから，本手法は実際の地形や人の移動を考慮に入れた密度クラスタリングをすることに対して有効であると考えられる．

6.3　定量的評価

6.3.1　実験

本項では図 12 に示す人工データに対して，EBSCAN と DBSCAN の両手法を用いてクラスタリングを行った．本論文では評価基準として，“purity” と “inverse purity” の調和平均である F 値を採用した．purity は出力されたクラスタの中に異なる正解クラスが存在しないかといった “純度” を測る指標であり，式 (3) で計算される．ここで，N はデータ数，K は正解クラスタの個数，L は生成されたクラスタ数，$n_{i,j}$ は生成された i 番目のクラスタにおいて j 番目の正解クラスに属するデータ数である．

$$purity = \frac{1}{N}\sum_{i=1}^{L}\max_{j}(n_{i,j}) \tag{3}$$

inverse purity は正解クラスが異なる予想クラスタに分

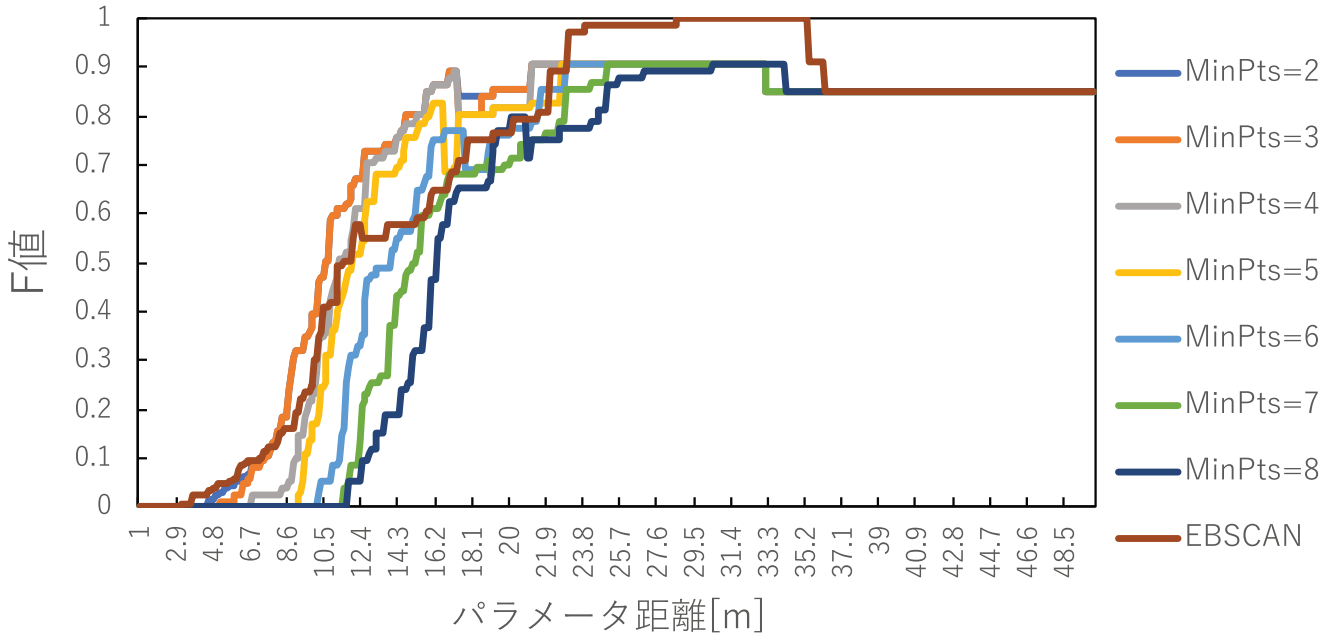

図 16　人工データに対する F 値
Fig. 16　F value for artificial data.

かれていないかを測る指標であり，式 (4) で計算される．

$$inverse\ purity = \frac{1}{N}\sum_{i=1}^{K}\left\{\frac{\max_{j}(n_{i,j})}{\sum_{i=1}^{K} n_{i,j}}\sum_{j=1}^{L} n_{i,j}\right\} \quad (4)$$

F 値は purity と inverse purity の調和平均であり，式 (5) で計算される．

$$F = \frac{2}{\frac{1}{purity} + \frac{1}{inverse\ purity}} \quad (5)$$

パラメータの設定範囲は EBSCAN の *toofar*，DBSCAN の ϵ ともに 1 m から 50 m の間で 0.1 m ずつ変化させた．また，今回用意した正解クラスタの数のうち，最も小さいクラスタの要素数は 8 であったため，DBSCAN の *MinPts* は 2 から 8 までの範囲で変化させた．この状態でクラスタリングを実行し，各パラメータにおける F 値の計算を行った．**図 16** に実験結果を示す．

6.3.2　考察

前項で行った実験結果に対する考察を行う．図 16 より EBSCAN は，DBSCAN と同様に，距離に関するパラメータが小さいうちは低い F 値をとり，パラメータの値が増加していくにつれて F 値は高くなることが分かる．この F 値の増加傾向はある程パラメータを増加させたところで頭打ちとなり，さらにパラメータを増加させると，F 値は減少する．こうした傾向の中で EBSCAN はつねに DBSCAN よりも高い F 値をとっているわけではない．しかし，測定した範囲において，DBSCAN における F 値の最高値は 0.907 程度であったのに対し，EBSCAN における F 値の最高値は 1 であった．

ここで，DBSCAN の F 値が 1 にならなかった理由に関する考察を**図 17** および，**図 18** を用いて行う．図 17 は DBSCAN で $MinPts = 3$，$\epsilon = 17.3$ m，図 18 は DBSCAN で $MinPts = 3$，$\epsilon = 17.4$ m に設定した場合のクラスタリング結果である．図 17 の時点では，作成した人工データ内の右下のクラスタが 2 つに分かれたうえ，クラスタに追加されていない要素があることが分かる．しかし，ϵ を大きくした図 18 では右下のクラスタの要素数不足やクラスタが分裂している状況を解決するより先に，左上のクラスタが結合していることが分かる．つまり，ϵ が小さい状態だと，1 つの正解クラスタが分裂してしまうが，ϵ を大きくすると異なる正解クラスタが結合した状態で結果が出力されてしまう．

一方，EBSCAN では，移動軌跡の交点に着目して距離の計算を行っている．図 12 を見て分かるように，左上に存在する 2 つのクラスタ間を直接結ぶような移動軌跡や，その交点といったものは存在しない．したがって，この 2 つのクラスタは別のクラスタとして出力することができ，F 値が 1 となったと考えられる．

また，図 16 において F 値が 1 となったのは瞬間的なものではなく，ある程度長い区間であったことから，パラメータ設定の難易度は *MinPts* を決め打ちした DBSCAN と比べても遜色がないと考えられる．さらに，EBSCAN では DBSCAN の *MinPts* に相当するパラメータは存在しないため，パラメータ設定の難易度は EBSCAN の方が容易であるといえる．これらのことから，適切なパラメータを設定することによって，EBSCAN は DBSCAN よりも精度の高いクラスタリングを行うことが可能であるといえる．

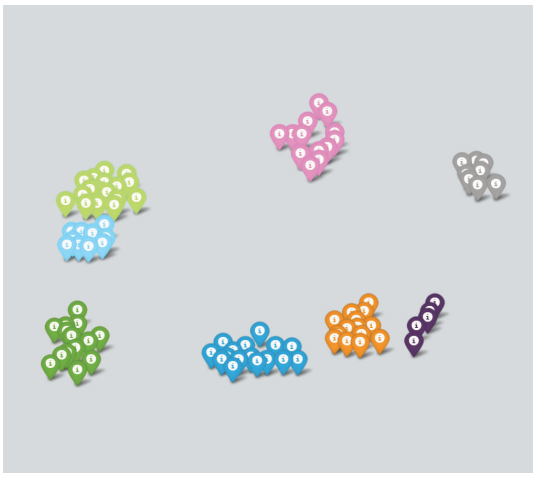

図 17　人工データに対する DBSCAN（$\epsilon = 17.3\,\mathrm{m}$，$MinPts = 3$）のクラスタリング結果

Fig. 17　Clustering result of DBSCAN ($\epsilon = 17.3\,\mathrm{m}$, $MinPts = 3$) for artificial data.

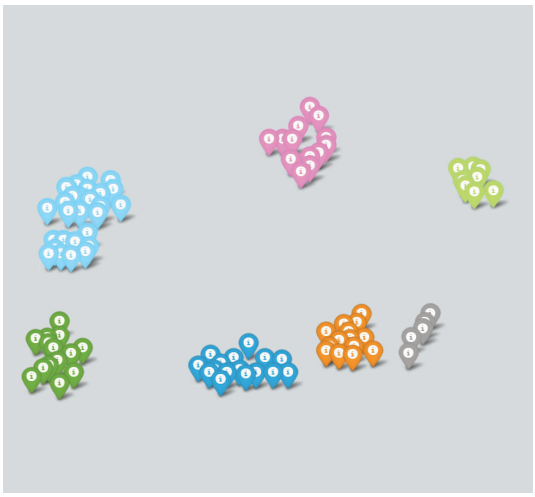

図 18　人工データに対する DBSCAN（$\epsilon = 17.4\,\mathrm{m}$，$MinPts = 3$）のクラスタリング結果

Fig. 18　Clustering result of DBSCAN ($\epsilon = 17.4\,\mathrm{m}$, $MinPts = 3$) for artificial data.

7.　まとめと今後の課題

本論文では，軌跡の絡まりに着目して時空間データの密度クラスタリングを行うアルゴリズム EBSCAN を提案した．既存手法である DBSCAN がパラメータを 2 つ必要とするのに対して，提案手法である EBSCAN は 1 つのみであり，適切なパラメータの選択が容易であるという特徴を持つ．

また，図 3 で示したような，サービスとしてビッグデータを提供する環境における，同時多発的に繰り返されるクエリに対して，高いスループットが得られることを実験にて示した．

さらに実データを使った実験から，通路が入り組んでいながらも 1 カ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ることを示した．この結果は EBSCAN が，実際の人の移動軌跡に即したクラスタを生成していることを示しており，実際の地形や移動を考慮した密度クラスタリングに有効であると考えられる．同時に，人工的に生成したデータに対する実験から，パラメータを適切に設定することで，EBSCAN は DBSCAN よりも精度の高いクラスタリングが可能であることを示した．

本論文では，利用したデータの特性上，移動軌跡をデータ間を直線で結ぶことで疑似的に生成しているが，実際の移動軌跡を本研究に適用させることは今後の課題である．また，本論文の時点ではデータが頻繁に更新されるような環境は考慮しておらず，インクリメンタルな更新への対応も今後の課題としている．

謝辞　本研究の一部は JSPS 科研費 JP19K11982 の助成を受けた．

参考文献

[1] Twitter, Inc.: Q3 2019 Letter to Shareholders, available from ⟨https://s22.q4cdn.com/826641620/files/doc_financials/2019/q3/Q3-2019-Shareholder-Letter.pdf⟩.
[2] Microsoft: IoT Signals: SUMMARY OF RESEARCH LEARNINGS 2019, available from ⟨https://azure.microsoft.com/mediahandler/files/resourcefiles/iot-signals/IoT-Signals-Microsoft-072019.pdf⟩.
[3] Sakaki, T., Okazaki, M. and Matsuo, Y.: Earthquake shakes Twitter users: Real-time event detection by social sensors, *Proc. 19th International Conference on World Wide Web*, pp.851–860, ACM (2010).
[4] Kan, Z., Tang, L., Kwan, M.-P., Ren, C., Liu, D. and Li, Q.: Traffic congestion analysis at the turn level using Taxis' GPS trajectory data, *Computers, Environment and Urban Systems*, Vol.74, pp.229–243 (2019).
[5] Shirai, M., Hirota, M., Yokoyama, S., Fukuta, N. and Ishikawa, H.: Discovering multiple HotSpots using geo-tagged photographs, *Proc. 20th International Conference on Advances in Geographic Information Systems*, pp.490–493 (2012).
[6] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise, *Kdd*, Vol.96, No.34, pp.226–231 (1996).
[7] Liu, J., Huang, Z., Chen, L., Shen, H.T. and Yan, Z.: Discovering areas of interest with geo-tagged images and check-ins, *Proc. 20th ACM International Conference on Multimedia*, pp.589–598 (2012).
[8] Hu, Y., Gao, S., Janowicz, K., Yu, B., Li, W. and Prasad, S.: Extracting and understanding urban areas of interest using geotagged photos, *Computers, Environment and Urban Systems*, Vol.54, pp.240–254 (2015).
[9] Yokoyama, S., Bogárdi-Mészöly, Á. and Ishikawa, H.: Ebscan: An entanglement-based algorithm for discovering dense regions in large geo-social data streams with noise, *Proc. 8th ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, p.7, ACM (2015).
[10] Nasibov, E.N. and Ulutagay, G.: Robustness of density-based clustering methods with various neighborhood

relations, *Fuzzy Sets and Systems*, Vol.160, No.24, pp.3601–3615 (2009).
[11] Dash, M., Liu, H. and Xu, X.: '1 + 1 > 2': Merging distance and density based clustering, *Proc. 7th International Conference on Database Systems for Advanced Applications, DASFAA 2001*, pp.32–39, IEEE (2001).
[12] Ankerst, M., Breunig, M.M., Kriegel, H.-P. and Sander, J.: OPTICS: Ordering points to identify the clustering structure, *ACM Sigmod Record*, Vol.28, No.2, pp.49–60 (1999).
[13] Esmaelnejad, J., Habibi, J. and Yeganeh, S.H.: A novel method to find appropriate ε for DBSCAN, *Asian Conference on Intelligent Information and Database Systems*, pp.93–102, Springer (2010).
[14] Karami, A. and Johansson, R.: Choosing dbscan parameters automatically using differential evolution, *International Journal of Computer Applications*, Vol.91, No.7, pp.1–11 (2014).
[15] Shi, J., Mamoulis, N., Wu, D. and Cheung, D.W.: Density-based place clustering in geo-social networks, *Proc. 2014 ACM SIGMOD International Conference on Management of Data*, pp.99–110, ACM (2014).
[16] Kisilevich, S., Mansmann, F. and Keim, D.: P-DBSCAN: A density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos, *Proc. 1st International Conference and Exhibition on Computing for Geospatial Research & Application*, p.38, ACM (2010).
[17] Bentley, J.L. and Ottmann, T.A.: Algorithms for reporting and counting geometric intersections, *IEEE Trans. Computers*, No.9, pp.643–647 (1979).
[18] Bartuschka, U., Mehlhorn, K. and Näher, S.: A robust and efficient implementation of a sweep line algorithm for the straight line segment intersection problem, *Proc. Workshop on Algorithm Engineering*, Citeseer (1997).
[19] Boissonnat, J.-D. and Preparata, F.P.: Robust plane sweep for intersecting segments, *SIAM Journal on Computing*, Vol.29, No.5, pp.1401–1421 (2000).
[20] Benouamer, M.O., Jaillon, P., Michelucci, D. and Moreau, J.-M.: A lazy solution to imprecision in computational geometry, *Proc. 5th Canad. Conf. Comput. Geom*, pp.73–78 (1993).
[21] Hubeny, K.: Zur Entwicklung der Gauss' schen Mittelbreitenformeln, *Österreichische Zeitschrift für Vermessungswesen*, Vol.42, No.1, pp.8–17 (1954).

伊藤 光太郎

東京都立大学大学院システムデザイン研究科博士前期課程在学．2020 年首都大学東京システムデザイン学部システムデザイン学科卒業．データマイニングの研究に従事．

横山 昌平　（正会員）

東京都立大学システムデザイン学部准教授．2006 年東京都立大学大学院工学研究科博士課程修了．博士（工学）．株式会社オリエンタルランド，産業技術総合研究所，静岡大学情報学部を経て 2018 年より現職．データ工学の研究に従事．

（担当編集委員　若林　啓）