

モバイル端末向けジオフェンシングにおける更新回数の削減

根本 潤^{1,a)} 遠山 元道^{2,b)}

受付日 2020年9月9日, 採録日 2021年1月11日

概要: モバイル端末などを活用した位置情報サービスでは、ユーザが関心のある特定の領域へ入ったことを検知するジオフェンシングが広く活用されている。モバイル端末におけるジオフェンシングでは、すべてのアプリケーションが平等に当該機能を使用できるように、監視対象の領域の数が20~100に制限されていることが一般的である。そのため、設定するジオフェンスを動的に更新しながら、任意の数のジオフェンスへの移動を漏れなく監視可能なサーバ協調型のジオフェンシングが提案されている。従来のサーバ協調型のジオフェンシングでは、ジオフェンスの決定方式として、四分木方式やナイーブな空円方式が提案されているが、モバイル端末の消費電力に影響するジオフェンスの更新回数に関して課題がある。そこで、本論文では、 k 個の近傍を監視対象としてジオフェンスを設定しつつ、それ以外の監視対象領域が存在しない大きな空円に制御用ジオフェンスを設定する k 近傍アウェア空円方式を提案し、ジオフェンスの更新回数を削減する。人々の1日の移動を表現したオープンデータであるOpenPFLOWを用いた大規模なシミュレーションにより、提案方式が、四分木方式に対して約42~76%、ナイーブな空円方式に対して約54~75%更新回数を削減できることを示す。

キーワード: 位置情報サービス, ジオフェンシング, GPS

Reducing Updates for Geofencing in Mobile Devices

JUN NEMOTO^{1,a)} MOTOMICHI TOYAMA^{2,b)}

Received: September 9, 2020, Accepted: January 11, 2021

Abstract: Geofencing that detects that users have entered a specific area of interest is widely used for location-based services. Geofencing in mobile devices generally has a limitation on the number of fences that can be registered due to the specification of the mobile device OSs. Therefore, server-assisted geofencing has been proposed that can monitor the movement to any number of geofences while dynamically updating the geofences that registered. The quadtree approach and the naive empty circle approach in conventional server-assisted geofencing have an issue regarding the number of geofence updates that affect the power consumption of mobile devices. In this paper, we propose a k -nearest-neighbors-aware empty circle approach that reduces the number of geofence updates. With a large-scale simulation using OpenPFLOW, we demonstrate that the proposed approach can reduce about 42 to 76% of geofence updates compared to the quadtree approach, and reduce about 54 to 75% compared to the naive empty circle approach.

Keywords: location-based services (LBS), geofencing, GPS

1. はじめに

モバイル端末などでユーザの位置情報を取得し、当該位置情報に基づいて各種サービスを提供する位置情報サービスでは、ユーザが特定の領域（以降、Area of Interest: AoIとも呼ぶ）へ入ったことを検知するジオフェンシングが広く活用されている。位置情報サービスにおけるジオフェン

¹ 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University,
Yokohama, Kanagawa 223–8522, Japan

² 慶應義塾大学理工学部
Faculty of Science and Technology, Keio University,
Yokohama, Kanagawa 223–8522, Japan

a) nemoto@keio.jp

b) toyama@ics.keio.ac.jp

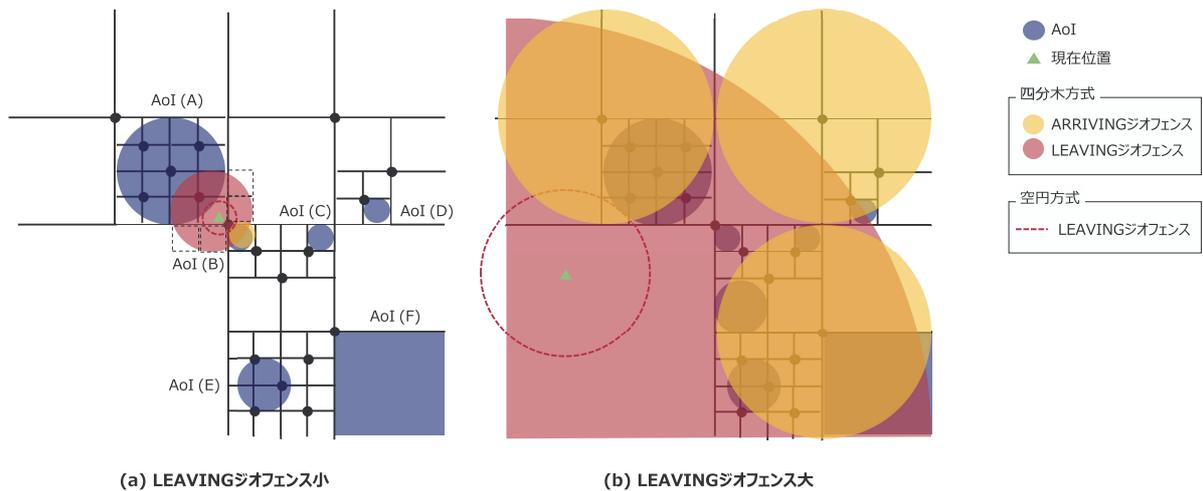


図 1 四分木方式と空円方式におけるジオフェンス設定例

Fig. 1 Examples of geofence settings with quadtree approach and empty circle approach.

シングの活用分野は、広告、観光、防災、感染症対策など多岐にわたる [1], [2], [3], [4].

モバイル端末向けの主要な OS である Android や iOS では、標準でジオフェンシング機能が提供されている。ただし、ジオフェンシングのために必要な現在位置の監視は、GPS などのハードウェアに依拠したものであり、モバイル端末においては、すべてのアプリケーションが平等に当該機能を使用できるように、監視対象の領域の数に制約がある [5], [6]。具体的には、1 アプリケーションにつき、Android で 100 個、iOS で 20 個に制限されている。

このような制約のため、設定対象の AoI が多数ある場合、モバイル端末上のアプリケーションで登録するジオフェンスを動的に更新しながら、任意の数のジオフェンスへの移動を漏れなく監視可能な方式が提案されている [2], [7], [8]。特に、文献 [7] や文献 [8] で提案されているような対象ジオフェンスの選択処理をサーバ側で実施する形態を、本論文では、サーバ協調型ジオフェンシングと呼ぶ。サーバ協調型ジオフェンシングでは、ジオフェンスの設定に必要な幾何学的な計算をサーバ側へオフロードできる。

従来のサーバ協調型のジオフェンシングでは、ジオフェンスの決定方式として、四分木方式 [7] や空円方式 [8] が提案されているが、ジオフェンスの更新頻度に関して課題がある。サーバ協調型ジオフェンシングでは、ユーザの移動に応じてジオフェンスを更新する際、ネットワーク経由で更新要求を送信するが、文献 [7] によれば更新要求の回数に対してほぼ線形にバッテリーが減少していくとが示されている。したがって、モバイル端末におけるジオフェンシングでは、消費電力を削減するため、ジオフェンスの更新回数削減が重要な課題である。

そこで、本論文では、ジオフェンスの更新回数を削減

するため、 k 近傍アウェア空円方式を提案する。 k 近傍アウェア空円方式（以下、 k 空円方式ともいう）では、現在位置に最も近い k 個の AoI を除いた状態で、それ以外の AoI が存在しない領域（空円）に対して制御用のジオフェンスを設定する。この制御用のジオフェンスを出した場合にのみ、サーバへ新しいジオフェンスを要求する。また、空円の算出方法としては、従来の DCC (Dynamic Centered Circle) や DSC (Dynamic Shifted Circle) [9] に加えて、現在位置を含み、かつ AoI が存在しない最大の空円である LEC (Largest Empty Circle) の利用について検討する。

本論文の構成は以下のとおりである。まず、2 章で関連研究について述べる。次に、3 章で提案する k 近傍アウェア空円方式によるサーバ協調型ジオフェンシングについて述べる。4 章では、提案方式の評価のために行った実験とその結果について述べる。そして、5 章で、提案方式の実用上の諸課題について議論したのち、6 章で結論を述べる。

2. 関連研究

四分木方式

Loyola らは四分木方式のサーバ協調型ジオフェンシングを提案している [7]。図 1 (a) を用いて四分木方式の概要を説明する。四分木方式では、まず、各 AoI を四分木データ構造にマッピングする。マッピングは、まず、最上位の矩形（最大サイズの矩形）が AoI を完全に包含しているかどうかを判定する。もし包含している場合、当該 AoI の一部としてその矩形をマッピングする。もし AoI と部分的に重なり合う場合には、その矩形を四分分割し、分割された矩形について再帰的に判定を繰り返す。図 1 (a) は、青い円と矩形で表される AoI を四分木にマッピングした状態を示している。AoI の形状に制約はなく円や多角形など任意の領

域を登録することができる。

四分木方式では、現在位置に対応する矩形を四分木をたどって取得し、当該矩形と同サイズの周囲8つの矩形*1にAoIがマッピングされているかを確認する。確認対象の矩形がAoIにマッピングされていて、かつ、そのAoIが現在いる矩形にマップされているものと異なる場合には、その矩形に内接する円をARRIVINGジオフェンスとして設定する。また、8つの確認対象矩形全体に内接する円をLEAVINGジオフェンスとして設定する。たとえば、図1(a)におけるAoI(B)上の黄色い円がARRIVINGジオフェンスとして設定されるが、ユーザはすでにAoI(A)にいるため、その矩形には設定されない。ユーザがARRIVINGジオフェンスへ到着するかLEAVINGジオフェンスを離れた場合には、サーバへ現在位置情報を引数として有するジオフェンス更新要求を送信する。サーバは、現在位置がAoIに入ったかを確認するとともに、新しいジオフェンスを計算して応答する。

四分木方式には2つの課題がある。1つは、ARRIVINGジオフェンスに到着した際にもジオフェンスの更新が発生するため、サーバへの要求が増大する点である。たとえば、図1(b)のように、ユーザがAoIがマッピングされていない大きな矩形にいる場合には、大きなARRIVINGジオフェンスが設定され、それらに到着した時点で再度、より小さいジオフェンスに更新される。Loyolaらが述べているように、ジオフェンスの更新回数はモバイル端末のバッテリー消費に直結するため、ジオフェンスの更新回数削減は重要な課題である。もう1つは、モバイル端末のGPS位置情報取得精度やジオフェンシングの精度によらず偽陽性および偽陰性が発生しうる点である。たとえば、図1(a)におけるAoI(E)のように、ジオフェンスを設定したい領域と矩形とのあいだにずれがある場合、周囲のいくつかの矩形に余分にAoIが設定されうるため、これが偽陽性の発生につながる。また、図1(b)において、四分木方式が設定するARRIVINGジオフェンスがAoIをカバーしきれていないことから分かるように、四分木の矩形サイズが異なる領域の移動の際に偽陰性が発生しうる。

空円方式

Garzonらは、AoIへの出入りの時系列的なつながりを考慮したGeofencing 2.0を提案している[8]。Geofencing 2.0のプロトタイプでは、現在位置を中心として最寄りのAoIまでの距離を半径とする円を、ジオフェンスの更新が不要な安全圏(LEAVINGジオフェンス)として設定する(図1における赤破線)。LEAVINGジオフェンスを離れた際には、サーバに問合せを行いAoI内に入ったか否かを判定する。AoIが存在しない(空の)円を用いたこの方式を、本論文では空円方式と呼ぶ。空円の算出方法として、

近接ユーザ検出に関するKüpperらの研究において、DCCとDSCの2つの方法が提案されており[9]、Garzonらは、DCCに基づく空円方式を採用している*2

Garzonらの空円方式は、LEAVINGジオフェンスのみを用いて、サーバ側でAoIへの出入りの判定を行うため、任意の形状のジオフェンスを取扱いが容易であり、かつ、アルゴリズムとしては四分木方式のように偽陰性が発生することはない。ただし、図1(a)のように、AoIが密集するような場所においては、空円が小さくなるため、ジオフェンスの更新回数が増大してしまうという課題がある。

k近傍方式

板崎らはあらかじめ定めた半径を持つ制御用ジオフェンス(LEAVINGジオフェンス)と、近傍AoIをモバイル端末OSの上限値一杯までARRIVINGジオフェンスとして設定する方式を提案している[2]。また、Mallikらは近傍AoIのみをARRIVINGジオフェンスとして設定し、所定の距離を移動した際にそれらを更新する方式を提案している[4]。これらの方式では、AoIへの移動を漏れなく検出するためには必然的にLEAVINGジオフェンスが小さくなるため、更新回数が増大する。

3. 提案方式

前章で述べた従来研究の課題をふまえ、本論文ではk近傍アウェア空円方式(k空円方式)を提案する。

3.1 概要

k空円方式では、現在位置から最も近いk個のAoIに対して、ARRIVINGジオフェンスを設定し、それ以外のAoIが存在しない領域(空円)に対してLEAVINGジオフェンスを設定する。ARRIVINGジオフェンスへの到着の際には、四分木方式のようにサーバへの問合せは不要で、モバイル端末側でAoIへの到着であると判断できる。LEAVINGジオフェンスから離脱した際には、サーバに問い合わせで新しいk個のARRIVINGジオフェンスと1個のLEAVINGジオフェンスを取得する。従来のナイーブな空円方式に比べて、近隣のAoIを除外した分だけLEAVINGジオフェンスが大きくなるため、サーバへの問合せ、ジオフェンスの更新を削減可能となる。

3.2 空円算出

k空円方式では、k近傍AoIを除外した後、どのような空円を設定するかについて選択肢がある。本論文では、LEAVINGジオフェンスを決定する際の空円の算出方法として、文献[9]で提案されているDCC、DSCに加えて、Largest Empty Circle(LEC)を検討する。図2にDCC、DSC、

*1 図1(a)における破線の矩形のように、四分木上で必ずしも同サイズで分割されているとは限らない。

*2 文献[8]の中ではDSCを修正したものであると言及されているが、現在位置を中心とした円であることから実質的にDCCであるといえる。

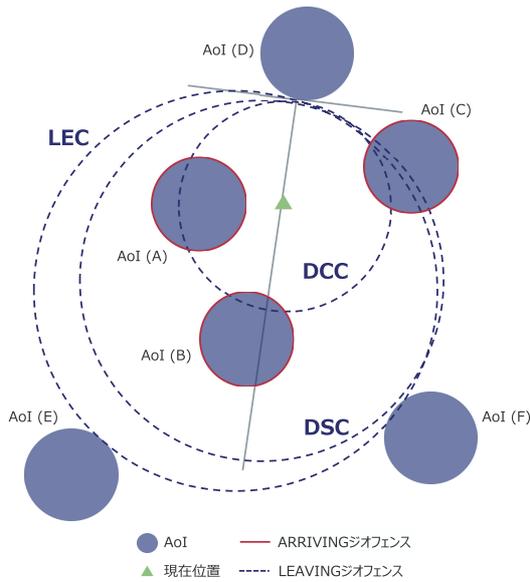


図 2 k 近傍アウェア空円方式

Fig. 2 K-nearest-neighbors-aware empty circle approach.

LECそれぞれの空円の比較を示す. 図 2 では, (A)~(F) 6 個の AoI が図示されており, k を 3 とした場合 (AOI(A)~(C) を除いた場合) の各空円算出結果の違いを示している.

k 空円方式において, DCC は, 現在位置を中心として $k+1$ 番目に近い AoI (図 2 では AoI(D) が該当) までの距離を半径とする円である. また, DSC は, DCC と AoI の接点と現在位置を結ぶ延長線上に中心を持ち, かつ, k 個の AoI 以外のいかなる AoI にも接しない最大の円である. DSC は DCC に比べて大きな空円を描くが, 多くの場合, それよりも大きな最大空円 (LEC) が存在するため, これを利用する.

3.3 アルゴリズム

本節では, LEC を用いた k 空円方式により, ジオフェンスを算出するアルゴリズムについて説明する. 簡単のため, はじめに, すべての AoI が半径 r_a の円である場合を例に説明した後, 任意の AoI への拡張方法について述べる. Algorithm 1 に, 算出アルゴリズムの疑似コードを示す.

アルゴリズムの入力は, 円形 AoI の中心を母点として作成されたボロノイ図 V , k 近傍探索に用いる索引木 T (本論文では, kd 木を利用), ユーザの現在位置 q , パラメータ k , 円形 AoI (=ARRIVING ジオフェンス) の半径 r_a である. ボロノイ図は, 同一距離空間上の各点がどの母点に近いかによって領域分割された図形であり, 最大空円を求めるために使用する.

まず, 索引木を用いて k 個の最近傍点を求める (1 行目). そして, 得られた k 個の最近傍点をボロノイ図から一時的に削除し, その状態で現在位置 q が含まれるボロノイセルを求める (2 行目). ボロノイセルは各母点どうしの二等分線の一部を辺 (ボロノイ辺) とする凸多角形であり, ボロ

Algorithm 1 ジオフェンス算出アルゴリズム

```

Input:  $V$ : voronoi diagram,  $T$ : index tree,  $q$ : query point,  $k$ :
number of of nearest neighbors,  $r_a$ : arriving geofences radius
Output:  $C_a$ : list of arriving geofences,  $C_l$ : leaving geofence
1: neighbors = kNearestNeighborSearch( $T$ ,  $q$ ,  $k$ )
2:  $f$  = getVoronoiCellWithUpdatedDiagram( $V$ , neighbors,  $q$ )
3:  $p$  = getSite( $f$ )
4: vertices = getVertices( $f$ )
5: max = -1
6: for all  $v$  in vertices do
7:    $r$  = getDistance( $v$ ,  $p$ )
8:    $d$  = getDistance( $v$ ,  $q$ )
9:   if  $d \leq r$  &&  $r > \text{max}$  then
10:     center =  $v$ 
11:     max =  $r$ 
12:   end if
13: end for
14:  $C_l$  = makeLeavingGeofence(center, max -  $r_a$ )
15: for all  $n$  in neighbors do
16:    $C_a$ .append(makeArriveGeofence( $n$ ,  $r_a$ ))
17: end for
18: return  $C_a$ ,  $C_l$ 
    
```

ノイセルから母点 p と, 凸多角形の頂点 (ボロノイ点) を得る (3~4 行目). ここで, ボロノイ点は q を含む最大空円の中心の候補であるため [10], 各ボロノイ点と母点 p および現在位置 q との距離をそれぞれ求め, 現在位置を含む最大の空円を得る (6~12 行目). そして, ARRIVING ジオフェンスの半径分だけ差し引いた距離を半径とする最大空円を LEAVING ジオフェンスとして作成する (14 行目). ARRIVING ジオフェンスについては, 先に取得した k 個の最近傍点それぞれを中心として作成する (15~17 行目).

なお, ARRIVING ジオフェンスの半径分を差し引く際に現在位置 q が LEAVING ジオフェンスの外に出るケースがある. その場合には, DSC に基づく空円算出へフォールバックする. 具体的には, 母点 p と現在位置 q を結ぶ半直線とボロノイ辺との交点を求めることで, DSC に基づく空円の中心が得られる.

以上が, 固定サイズの円形 AoI に対するジオフェンス算出アルゴリズムである. 任意のサイズの円形 AoI に対してジオフェンスを設定する場合には, 円に対するボロノイ図である加法的重み付きボロノイ図を用いることで, 同様に最大空円を求めることができる. さらに, 円形以外の AoI をカバーするジオフェンスを設定する場合には, そうした任意のサイズの円形ジオフェンスを複数並べることを想定している.

4. 評価

4.1 評価環境

提案方式の適用により, ジオフェンスの更新回数が削減されることを確認するため, 人々の移動に関するオープンデータである OpenPFLOW [11] を用いてシミュレーショ

表 1 各シナリオの統計情報

Table 1 Statistical information for each scenario.

	駅シナリオ	CVS シナリオ
移動レコード数	162,323,118	58,745,086
エージェント数	2,166,201	956,862
平均移動レコード数	75	61
平均 AoI 訪問回数	12.5	4.7 (r=50m) 15.5 (r=100m)

ン評価を行った。OpenPFLOW は、朝自宅から通勤・通学し、夜帰宅するような人々の 1 日の移動を表現した GPS ログのようなデータであり、エージェントと呼ばれる抽象化された人物の 1 分ごとの緯度経度が移動手段とともに提供されている*3。

駅シナリオ

設定するジオフェンスとしては、2つのデータセットを用いた。1つは、各ジオフェンスが比較的疎に配置されることが期待される駅データであり、国土交通省が提供する GIS データである「国土数値情報」を使用した [12]。対象駅は、OpenPFLOW におけるエージェントが移動する緯度経度範囲にあるすべての駅、2,264 駅である。これらの駅に対し駅全体を覆う半径 300m のジオフェンスを設定する。

コンビニエンスストア (CVS) シナリオ

もう一つのデータセットは、比較的密にジオフェンスが配置されることが期待されるコンビニエンスストア (CVS) データ [13] であり、対象は後述する矩形領域に含まれる 4,311 店である。コンビニエンスストアについては、全体を覆うと期待できる半径 50m と、Android におけるジオフェンスの最小半径の推奨値である 100m の 2 通りについて評価した。

前者の駅データを用いたシナリオでは、OpenPFLOW で提供されている全 8.2 億件のデータから移動中とマークされた全レコード約 1.6 億件を使用した。後者の CVS データを用いたシナリオでは、ジオフェンスの更新が多発しうる条件下での評価を行うため、東京 23 区を全体を覆う矩形*4内における移動レコード約 0.6 億件を使用した。これらの正確な値と、ユニークなエージェントの数、各エージェントが平均何回 AoI を訪問したかといった統計情報を、表 1 に示す。

シミュレータ

評価に使用するシミュレータは、計算幾何学向けライブラリである CGAL [14] を使用し、C++を用いて実装した。具体的には、k 近傍探索のための kd 木クラスや、ポロノイ図クラスを使用して、空円方式を実装している。四分木に関しては CGAL に該当クラスがないため、スクラッチで実装した。また、文献 [7] における四分木方式では移動速

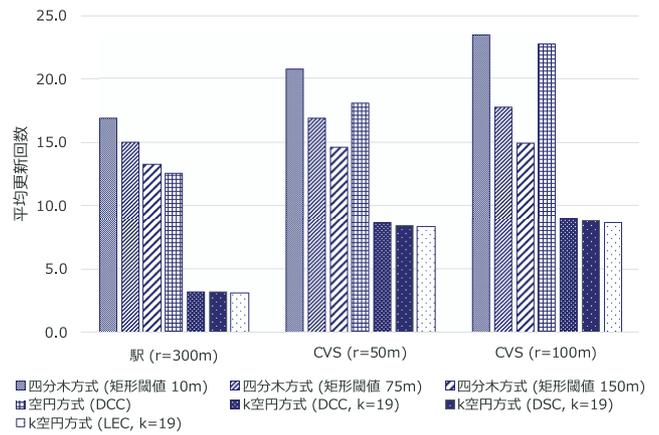


図 3 各シナリオにおける更新回数の比較

Fig. 3 Comparison of number of updates for each scenario.

度を入力に組み込んでいるが、シミュレータでは現在位置のみに基づいてジオフェンスを設定した。この変更は更新回数に影響するものの、小さな LEAVING ジオフェンスが設定が回避されるため、四分木方式に不利に働くことはないと考えている。なお、提案方式については、Algorithm 1 のうち、ポロノイ図の動的な更新が未対応であり、つどポロノイ図を再作成している。これについては、ジオフェンスの算出時間に影響するものの、更新回数には影響しない。従来方式と提案方式のジオフェンス算出時間の比較については後述する。

4.2 従来方式との比較

図 3 は、各シナリオにおいて、エージェントごとのジオフェンスの平均更新回数を方式ごとに比較したものである。四分木方式については、矩形が一定サイズを下回るまで領域分割を行うが、その閾値は更新回数や精度に影響するため、10m、75m、150m の 3 つのパターンで評価した*5。また、k 空円方式のパラメータ k は、モバイル端末の主要 OS である Android、iOS のうち、ジオフェンス数の制約がより厳しい iOS の 20 個から、LEAVING ジオフェンスを 1 つ差し引いて k=19 とした。

図 3 に示すように、四分木方式は矩形閾値を大きくするほど更新回数は削減されていくが、k 空円方式 (LEC) は、四分木方式の最良の場合 (矩形閾値 150m) に対して、約 42% (CVS シナリオ) ~ 約 76% (駅シナリオ) 少ない更新回数となった。また、ナイーブな空円方式は四分木方式と同等かそれ以上の更新回数であり、k 空円方式は、それに対して約 54% (CVS シナリオ) ~ 約 75% (駅シナリオ) 少ない更新回数となった。一方、k 空円方式における空円算出方法に関しては、DCC と LEC で約 2.4~3.9%、DSC と LEC で約 0.2~1.6%と僅かな差であった。シナリオ間の違

*3 <https://github.com/sekilab/OpenPFLOW>

*4 北緯 35.53~35.82 度, 東経 139.56~139.93 度を使用

*5 領域分割の結果、実際の最小矩形サイズは、駅シナリオでそれぞれ約 17m, 133m, 266m, CVS シナリオで約 16m, 131m, 262m であった。

表 2 LEAVING ジオフェンスの半径 (m)
Table 2 Radius of LEAVING geofence (m).

方式	駅データ (300 m)	CVS データ (50 m)	CVS データ (100 m)
四分木方式 (矩形閾値 150 m)	788	577	563
空円方式 (DCC)	702	262	179
k 空円方式 (DCC, k=19)	4,274	1,001	941
k 空円方式 (DSC, k=19)	4,458	1,046	987
k 空円方式 (LEC, k=19)	5,275	1,117	1,061

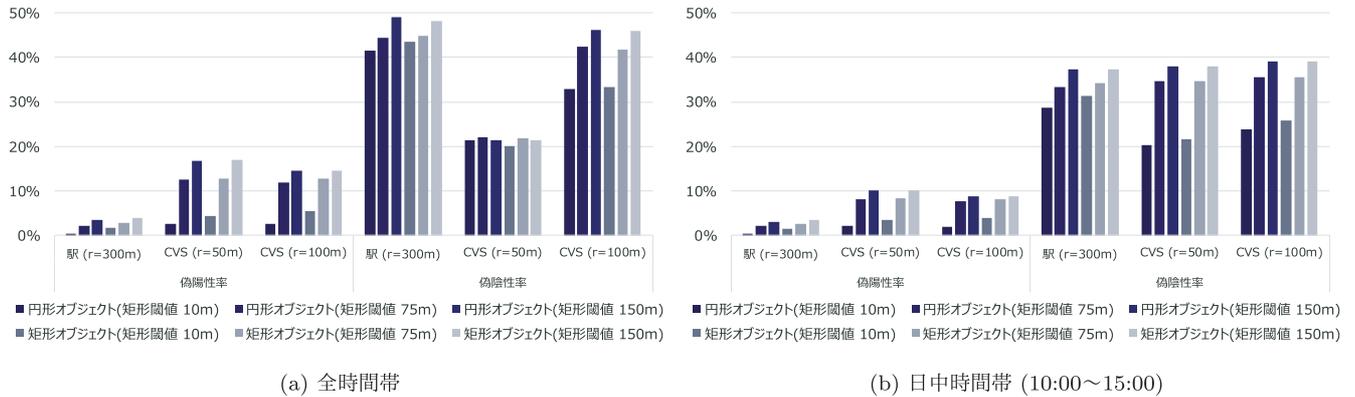


図 4 四分木方式における偽陽性率と偽陰性率
Fig. 4 False positive rate and false negative rate in quadtree approach.

いで見ると、各方式ともにジオフェンスの配置が密になるほど更新回数が増加する傾向にあり、従来方式と k 空円方式は、描ける空円の大きさが近づいていくことから、更新回数の差も小さくなった。

それぞれの方式の実際の LEAVING ジオフェンスの半径を表 2 に示す。表に示すとおり、従来方式と k 空円方式でそのサイズに顕著な差がある一方、k 空円方式間では DCC と LEC で約 10~20%の差に留まっている。

四分木方式における偽陽性と偽陰性

2 章で言及したように、四分木方式は仕組み上、偽陽性および偽陰性が発生しうる。そこで、四分木方式における矩形サイズがそれらに与える影響について評価した。なお、OpenPFLOW のデータは、朝自宅から通勤・通学し、夜帰宅するような人々の 1 日の移動を表現しているため、自宅などの周囲に AoI が存在しないエリア (領域分割がされていない大きな矩形のエリア) から、AoI に近づくことで、ARRIVING ジオフェンスでカバーされない領域へ入る確率が高くなってしまふと考えられる。この影響を確認するため、全時間帯のデータに加えて、一般的な通勤・通学の時間帯のレコードを除去した日中のみ (10:00~15:00) のデータについても評価を行った。図 4 の (a), (b) は、それぞれ、全時間帯のデータで評価した場合と日中のデータで評価した場合の各シナリオにおける矩形閾値ごとの偽陽性率、偽陰性率を示している。

偽陽性は、シミュレーション実行時、ARRIVING ジオフェンス外にいるにもかかわらず、四分木上でオブジェク

トが登録された矩形に位置していた場合を計上した。偽陰性は、空円方式で計上した正しい AoI 訪問回数から四分木方式における AoI 訪問回数と偽陽性数の差を差し引いて求めた。なお、円形ジオフェンスを四分木に登録したことで不当に偽陽性・偽陰性が増えていないかを確認するため、円形ジオフェンスに外接する矩形をジオフェンスとして登録した場合についても同様の実験を行った。

図 4 のとおり、円形・矩形にかかわらず矩形閾値が大きくなるにつれて同じような傾向で偽陽性、偽陰性が増加しているのが分かる。したがって、四分木方式は更新回数を削減するために矩形サイズ大きくすることはジオフェンシングの精度とトレードオフの関係にある。また、偽陰性の割合が全体として、20%強~50%弱と高い結果となった。事前に想定したように、通勤・通学時には、領域分割されていない大きな矩形のエリアから AoI に近づくことが多かったために検出が漏れたと考えられるが、日中のみのデータにおいても依然として 20%強~40%弱の偽陰性率であった。よって、四分木に複数の矩形サイズが混在する以上は、検出漏れを完全に防ぐことは困難であると考えられる。

移動手段別の考察

OpenPFLOW のデータは移動手段 (徒歩、自動車、鉄道、自転車) でラベル付けされており、移動手段ごとに各方式の更新回数の差に違いがあるかについても分析した。その結果、鉄道移動以外については図 3 と似た傾向であったが、CVS シナリオかつ鉄道移動のケースで、四分木方式およびナイーブな空円方式と、k 空円方式との差が約 20%程

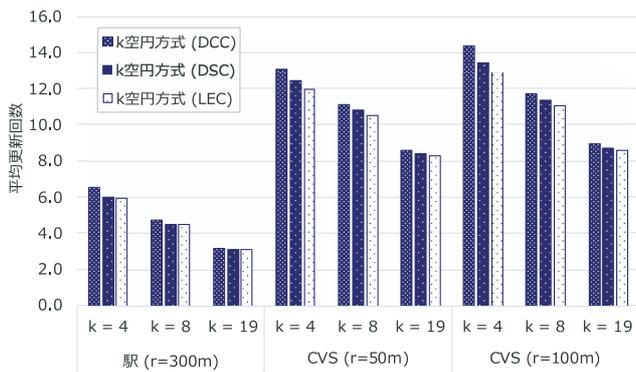


図 5 パラメータ k の更新回数への影響

Fig. 5 Effects of parameter k in number of updates.

度と他のケースに比べて小さかった。これは、空円が小さく移動速度が速いため、多くの判定で移動のたびに更新が発生した結果、両者の差が小さくなったと考えられる。空円の半径が 1km の場合に、列車が時速 60km 以上だとすると、OpenPFLOW のデータの間隔である 1 分後には、毎回空円を出ていることになるためである。

4.3 空円算出方法の比較

次に空円算出方法とパラメータ k の選択が更新回数に及ぼす影響を評価した。図 5 は、パラメータ k を 4, 8, 19 と変化させた際の、空円算出方法ごとの更新回数を示している。図に示すとおり、k が大きくなるほど更新回数は減少している。k=4 と k=19 で比較すると、駅シナリオで約 47~51%、CVS シナリオで約 31~38% 減少している。一方、空円算出方法の違いの影響は双対的に小さく、DCC と LEC は、最大で約 10% (駅シナリオ, k=4 の場合) 減少したが、最小で約 2.4% (駅シナリオ, k=19 の場合) の減少に留まった。DSC と LEC の差はさらに小さいが、これは 3.3 節で述べたようなフォールバック処理によるものではなく、DSC と LEC でほぼ同じような LEAVING ジオフェンスが設定されたためと考える。実際、k=19 の場合で、全更新回数のうちフォールバックに入ったのは、0.2~0.5% 程度であった。

上記結果から、基本的にはモバイル端末 OS の制約の上限から 1 引いた値を k (iOS で k=19, Android で k=99) として選べばよいと、DCC で十分な更新回数削減効果が期待できる。アプリケーションが、動的更新対象のジオフェンスのほかに別途静的なジオフェンスを必要とするなど、より小さい k を設定する必要がある場合には、LEC の利用が有用となる。

4.4 ジオフェンス算出時間の比較

モバイル端末からのジオフェンスの更新要求を受け、サーバ側で新しいジオフェンスを算出する時間について各方式間で比較評価を行った。本評価に用いたマシンは、CPU が

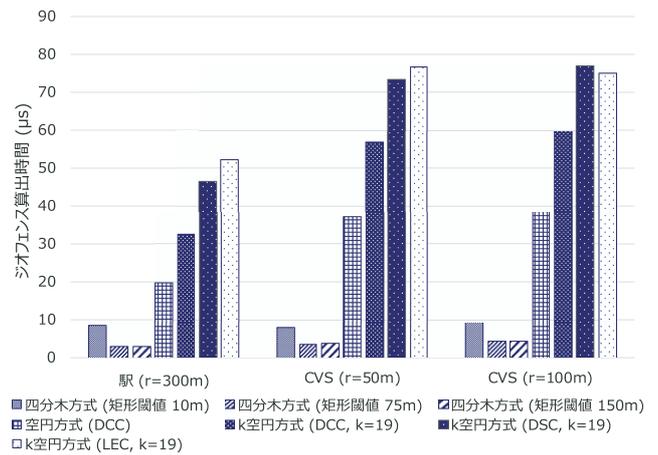


図 6 ジオフェンス算出時間の比較 (方式間)

Fig. 6 Geofence calculation time (among approaches).

Intel Xeon E5645 2.40 GHz×2, メモリが 192GB, OS が Ubuntu 18.04 (Linux Kernel 4.15.0) である。

図 6 は、OpenPFLOW のデータのうち 100 万件を用いて、各方式でジオフェンスを算出した際の更新 1 回あたりの平均処理時間を示している。なお、先述のように、シミュレータにおける k 空円方式はボロノイ図の動的更新に未対応であり、ボロノイ図の新規作成に平均 7.7~12.7ms 要したが、その他の処理の差異を明確化するため、図 6 では当該時間を除外している。文献 [15] によれば、ボロノイ図における母点の削除は $O(m \log m)$ (m は母点に隣接する近傍の数) の計算量で実行可能である。また、同文献では、CGAL ベースの試作で数マイクロ秒のオーダーで母点の削除が実行できていることから、提案方式においても数十~数百マイクロ秒で k 個の母点を削除できると考えている。

図 6 のように、ジオフェンス算出時間は、四分木方式が最も高速で、次いでナイーブな空円方式 (DCC) が高速であった。これは、四分木のノード検索と、kd 木による最近傍探索の計算量の差異が要因であると考えられる。また、ナイーブな空円方式と k 空円方式の差は、k 近傍探索コストに加え、ボロノイセルの辺や頂点をたどって空円の中心を探索コストが反映されたと考えられる。ただし、空円算出方法を問わず最大でも 80 マイクロ秒弱の算出時間であり、ボロノイ図の更新時間を考慮したとしても、数百マイクロ秒のオーダーになると考えられる。したがって、1 回のジオフェンス算出時間としては、ユーザの移動速度を鑑みて当該時間中に位置が数メートル単位で変わるわけではないため、ジオフェンシングの精度や品質に影響するものではないと考えている。一方で、多数のユーザを考えた場合、サーバ側の計算コストの観点では課題であり、詳細は 5 章で議論する。

次に、提案方式における各空円算出方法について、パラメータ k を変化させた際の、ジオフェンス算出時間の比較を図 7 に示す。各方式、各シナリオともに、パラメータ k

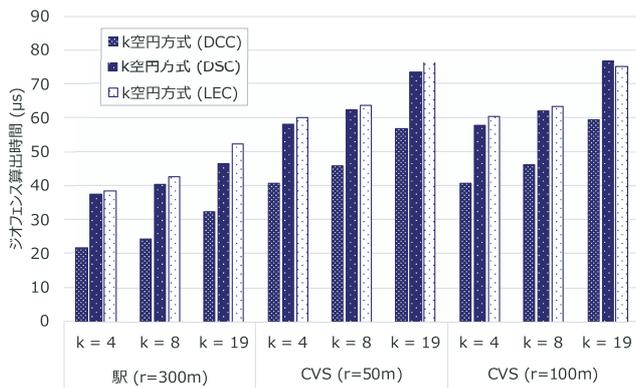


図 7 ジオフェンス算出時間の比較 (パラメータ k 間)

Fig. 7 Geofence calculation time (varying parameter k).

の増加にともなって、k 近傍探索コストが増加し、実行時間が伸びているのが分かる。

k=19 の場合、k=4 の場合に対して約 23~48%算出時間が増加しており、モバイル端末側の更新要求回数削減とはトレードオフの関係にある。また、LEC は DCC に対して約 25~77%算出時間が増加しており、2.4~10%の更新削減効果に対してやや高コストであるといえる。

5. 議論

本章では提案方式の実用化に向けて留意すべき事項について議論する。

5.1 任意の形状の AoI

本論文では各 AoI が円形でかつ半径が一定であるという条件下で評価を行ったが、現実には各 AoI の半径が異なるケースや、AoI が円形でないというケースがしばしばありうるため、その影響について考察する。

まず、各 AoI が異なる半径を有する円の場合であっても、ジオフェンス算出アルゴリズムとしては大きな違いはない。DCC による k 空円方式では、一定半径の場合と同様に、索引木を用いて (k+1) 近傍を求め、その円までの距離を半径とすればよい。DSC・LEC の場合は、先に述べたように、加法的重み付きボロノイ図を作成して空円を得る。更新回数についても AoI 配置関係が変わるのみで、従来方式に対して大きな空円が設定されることには変わりないため、依然として提案する k 空円方式の優位性に大きな違いはないと考える。

多角形など任意の形状の AoI については、複数の異なるサイズの円形ジオフェンスを隙間なく並べることで、上記のジオフェンス算出アルゴリズムをそのまま流用可能である。ただし、更新回数については、円形の AoI が密集することになるため、4 章の評価における CVS シナリオかそれ以上に従来方式との差は小さくなると予想される。また、円形ジオフェンスのサイズや敷き詰め状況によっては、提案方式においても偽陽性、偽陰性が発生することになる。

5.2 サーバ側の計算コスト

4 章で述べたように、提案する k 空円方式は、四分木方式に比べてサーバ側の計算コストが大きい一方で、ジオフェンスの更新回数が少なく、偽陽性や偽陰性が発生しない。本節では、計算コスト、更新回数、検出精度の 3 点をふまえて、提案方式の適用範囲について議論する。

k 空円方式は四分木方式に対して、約 42~76%ジオフェンスの更新回数を削減するが、その割合以上にジオフェンス算出時間が増大しており、全体としてサーバ側で必要となる計算コストは増加してしまう。したがって、サーバ側の計算コストを優先したい場合、提案方式の採用は不適である。しかし、防災や感染症対策といった安心・安全に関わるユースケース [3], [4] で AoI への侵入検知漏れを確実に排除したい場合には、四分木方式は不適であり、偽陰性の発生しない空円方式を採用する必要がある。

偽陰性の発生しない空円方式間の比較では、k 空円方式が約 54~75%更新回数を削減できる一方、1 回あたりのジオフェンス算出時間は、ナイーブな空円方式が、k 空円方式 (DCC, k=19) に対して約 35~39%短い。これにボロノイ図の更新時間も考慮すると、k 空円方式において、全体としてサーバ側で必要となる計算コストはナイーブな空円方式以上となりうる。したがって、k 空円方式におけるボロノイ図の更新時間の評価や、計算コスト削減は今後の課題である。

5.3 検出遅延

文献 [2] や文献 [8] で報告されているように、実システムにおいては、LEAVING ジオフェンスを出た際のモバイル端末側の検出・通知の遅れが課題となっている。提案方式においても、文献 [8] で述べられている手法と同様に、検出の遅延を考慮して一定距離 LEAVING ジオフェンスを小さくする必要があると考える。その場合、更新回数が増加することになるが、他の方式も同様の対処が必要であることをふまえると、その差は大きく変わらないと考える。

6. おわりに

本論文では、モバイル端末向けのジオフェンシングにおいて、端末の消費電力やサーバ側の処理負荷に影響するジオフェンスの更新回数を削減する方式について提案した。提案方式は、k 個の最近傍を監視対象として ARRIVING ジオフェンスを設定しつつ、それ以外の監視対象領域が存在しない大きな空円に制御用の LEAVING ジオフェンスを設定する k 近傍アウェア空円方式である。また、本論文では、監視対象領域が存在しない空円の算出方法として、従来研究で用いられている DCC, DSC に加えて、それらよりも大きい最大の空円 (LEC) の利用について検討した。

人々の 1 日の移動を表現したオープンデータである OpenPFLOW を用いた大規模なシミュレーションにより、

提案方式が、四分木方式に対して約 42~76%, ナイブな空円方式に対して約 54~75%更新回数を削減できることを示した。また、空円算出方法について、モバイル端末 OS の上限値一杯までジオフェンスを設定可能な場合では、DCC で十分な更新回数削減効果が期待でき、アプリケーションの制約などでジオフェンスの数が少数に制限される場合においては、LEC の利用が有用であることを示した。

モバイル端末における実際のジオフェンシングの挙動や精度に即した検出漏れの少ない LEAVING ジョフェンスの設定や、実システム上での実験、ジオフェンスの計算コスト削減が今後の課題である。

参考文献

[1] Bareth, U., Küpper, A. and Ruppel, P.: geoXmart - A Marketplace for Geofence-Based Mobile Services, *2010 IEEE 34th Annual Computer Software and Applications Conference*, pp.101-106 (2010).

[2] 板崎 輝, 渡邊悠太, 宇式一雅, 藤井 彰, 三宅正史: 観光向けアプリケーションを想定した適応的ジオフェンス制御, 研究報告デジタルコンテンツクリエーション 2 (2018).

[3] 篠原雅貴, 田島誠也, 日向 慧, 飯塚直亮, 齊藤圭世, 柴原直也, 高橋洗人, 岩井将行: 災害時に適応した行動を可能にするジオフェンスチェックラリアアプリケーションの開発, 第 78 回全国大会講演論文集, Vol.2016, No.1, pp.999-1000 (2016).

[4] Mallik, R., Hazarika, A.P., Ghosh, S., Sing, D. and Bandyopadhyay, R.: Development of an Android application for viewing Covid-19 containment zones and monitoring violators who are trespassing into it using Firebase and Geofencing, *Transactions of the Indian National Academy of Engineering*, Vol.5, pp.163-179 (2020).

[5] Google LLC: Create and monitor geofences — Android Developers (online), available from <https://developer.android.com/training/location/geofencing> (accessed 2020-09-10).

[6] Apple Inc.: Monitoring the User's Proximity to Geographic Regions — Apple Developer Documentation (online), available from https://developer.apple.com/documentation/corelocation/monitoring_the_user_s_proximity_to_geographic_regions (accessed 2020-09-10).

[7] Loyola, L., Wong, F., Pereira, D. and Sanson, H.: Extending Battery Lifetime of Mobile Devices with Geofence Capabilities on Dynamic-Speed Urban Environments, *Proc. 2nd ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, MobiGIS '13*, pp.51-58 (2013).

[8] Rodriguez Garzon, S. and Deva, B.: Geofencing 2.0: Taking Location-Based Notifications to the next Level, *Proc. 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '14*, pp.921-932 (2014).

[9] Küpper, A. and Treu, G.: Efficient Proximity and Separation Detection among Mobile Targets for Supporting Location-Based Community Services, *SIGMOBILE Mob. Comput. Commun. Rev.*, Vol.10, No.3, pp.1-12 (2006).

[10] Berg, D., Van Kreveld, M., Overmars, M., Schwarzkopf, O. and Cheong, O.: *Computational Geometry: Algorithms and Applications*, Springer (2000).

[11] Kashiya, T., Pang, Y. and Sekimoto, Y.: Open PFLOW: Creation and evaluation of an open dataset for typical people mass movement in urban areas, *Transportation Research Part C: Emerging Technologies*, Vol.85, pp.249-267 (2017).

[12] 国土交通省: 国土数値情報ダウンロードサービス (オンライン), 入手先 <https://nlftp.mlit.go.jp/ksj/> (参照 2020-09-10).

[13] 高橋三雄: 連載「フリーソフトによるデータ実践 GIS」東京都コンビニエンスストアの緯度経度データ, 統計情報研究開発センター (オンライン), 入手先 <https://www.sinfonica.or.jp/kanko/estrela/refer/s29/> (参照 2020-09-10).

[14] The CGAL Project: The Computational Geometry Algorithms Library (online), available from <https://www.cgal.org/> (accessed 2020-09-10).

[15] Dinis, J. and Mamede, M.: Updates on Voronoi Diagrams, *2011 8th International Symposium on Voronoi Diagrams in Science and Engineering*, pp.192-199 (2011).



根本 潤 (学生会員)

平成 16 年慶應義塾大学経済学部卒業。平成 18 年同大学大学院理工学研究科修士課程修了。同年株式会社日立製作所入社, 主にストレージシステムの研究開発に従事。平成 26 年 Carnegie Mellon University 客員研究員。平成 31 年同社退職。現在, 慶應義塾大学大学院理工学研究科博士課程在学中。主にデータベースの研究に従事。



遠山 元道 (正会員)

昭和 54 年慶應義塾大学工学部管理工学科卒業。昭和 59 年同大学大学院博士課程修了後, 管理工学科助手, 専任講師, 同情報工学科准教授を経て現在, 慶應義塾大学理工学部情報工学科教授。博士 (工学)。平成 8 年 Oregon Graduate Institute 客員研究員。平成 10~13 年科学技術振興事業団さきがけ研究 21「情報と知」領域研究員。主にデータベースの研究に従事。電子情報通信学会, 日本ソフトウェア科学会, IEEE, ACM 各会員。

(担当編集委員 平松 薫)