修士論文

複数スマートフォンのMACアドレスランダム化の非同期性を用いたODデータ自動取得手法の提案と評価

川島 将渡
奈良先端科学技術大学院大学
先端科学技術研究科
情報理工学プログラム

主指導教員: 藤川 和利 教授
情報基盤システム学研究室（情報科学領域）

令和4年2月24日提出

本論文は奈良先端科学技術大学院大学先端科学技術研究科に
修士 (工学) 授与の要件として提出した修士論文である。

川島 将渡

審査委員：

藤川 和利 教授	（主指導教員）
安本 慶一 教授	（副指導教員）
新井 イスマイル 准教授	（副指導教員）

複数スマートフォンのMACアドレスランダム化の非同期性を用いたODデータ自動取得手法の提案と評価*

川島 将渡

内容梗概

路線バスの運行効率を高めるには，バス利用者の利用動向を示すODデータの活用が必要不可欠であり，人手に頼らず取得する手法が望まれる．そのため，本研究ではODデータの自動取得を目的とし，バスの利用者数，金銭コスト，決済方法の観点から，スマートフォンが発するパケットを用いたODデータの自動取得手法について着目した．この手法は，バスの利用者数によらずODデータを取得可能であり，機材の導入にかかる金銭コストが小さいためである．しかし，プライバシー保護の観点から，ランダムMACアドレスが使用されるようになって以降，MACアドレスをランダム化するスマートフォンに対応するODデータ自動取得手法は確立されていない．MACアドレスランダム化に対応する手法として，スマートフォンが発するパケットからMACアドレスと識別子となる一定時間変更されないビット列を抽出し，MACアドレスが変更されたとき，識別子を基に新しいアドレスに更新するアドレス・キャリーオーバー・アルゴリズムがある．予備実験の結果からRSSIを識別子として用いることにしたが，バス車内を想定した場合，隣接した利用者のRSSIが同一値となり識別子が衝突するといった問題が考えられる．そこで，本研究では，複数のスマートフォンのMACアドレスランダム化の非同期性に着目し，変更後のMACアドレスの候補を制限することでODデータの取得精度を向上させるODデータ自動取得手法を提案する．京都府京都市の路線を対象に評価実験を行った結果，ODデータの取得精度は最も高い場合に63%，最も低い場合に20%となった．この結果は，既存手法がラン

*奈良先端科学技術大学院大学 先端科学技術研究科 修士論文, 令和4年2月24日.

ダム MAC アドレスを用いるスマートフォンを追跡できなかったのに対して本提案は追跡できることを示した．さらに，路線バスの通過バス停を除外した場合，OD データの取得精度は最も高い場合に 100%，最も低い場合に 60%と向上した．

キーワード

Bluetooth，MAC アドレス，OD 推定，公共交通

Proposal and Evaluation of Automatic OD Data Acquisition Method by Using Asynchronous MAC Address Randomization for Multiple Smartphones*

Masato Kawashima

Abstract

In order to improve the operational efficiency of bus routes, it is essential to utilize OD data, which indicates the usage trends of bus users, and a method of acquiring OD data that does not depend on human resources is desired. In this study, we focused on the automatic acquisition of OD data using packets sent from smartphones from the perspective of the number of bus users, money costs, and the items carried by the users. The reason for this method is that OD data can be obtained regardless of the number of bus users and the financial cost for installing the equipment is small. However, since random MAC addresses are used for privacy protection, an automatic OD data acquisition method using random MAC addresses has not been established for smartphones. To cope with the randomization of MAC addresses, the address carryover algorithm has been proposed that extracts the MAC address and identifier from the packets sent by the smartphone and updates the address to a new address based on the identifier when the MAC address is changed. However, in the case of a route bus, the RSSI of neighboring users has the same value, which may cause a problem of

*Master's Thesis, Graduate School of Science and Technology, Nara Institute of Science and Technology, February 24, 2021.

collision of identifiers. In this study, we focus on the asynchronous nature of the randomization of MAC addresses of multiple smartphones and propose an automatic OD data acquisition method that improves the accuracy of OD data acquisition by limiting the number of candidate MAC addresses after the change. As a result of an evaluation experiment on routes in Kyoto City, Kyoto Prefecture, Japan, the accuracy of OD data acquisition was 63% in the highest case and 20% in the lowest case. This results show that the proposed method can track smartphones that use random MAC addresses, whereas existing methods cannot track smartphones that use random MAC addresses. Furthermore, when bus stops passing by were excluded, the accuracy of OD data acquisition was 100% in the highest case and 60% in the lowest case.

Keywords:

Bluetooth, MAC address, OD estimation, Public transportation

目 次

図 目 次

表 目 次

1. はじめに

公共交通機関は毎日の通勤や通学，観光など様々な用途で利用されており，人々の移動手段として重要な役割を果たしている．特に路線バスは，地方において，市街地にある病院への通院やスーパーマーケットへの買い物，通学時の鉄道の乗り継ぎのための交通手段として重要である．

国土交通省の資料において，路線バス事業が抱える問題が以下のように挙げられている [1]．1つ目は，地方の路線バス利用者数の減少による採算悪化により，路線バス事業者の69%が赤字となっている点である．2つ目は，路線バスを運転する運転手の確保が困難となっている点である．路線バスの運転手は，長時間労働かつ年間所得額が低いため，若年者に就職先として敬遠されている．

こうした現状の路線バス事業において，できる限り少ない労働力と車両数で路線バスを運行できるようにダイヤを調整し，採算性を向上させることが重要となる．その目的のために，路線バス事業者は，各バス停間での路線バス利用者数といった利用動向を的確に把握することに努めている．利用動向を把握することができれば，乗り換えをする利用者が多い路線での乗り換え不要な路線の追加や利用者が多いバス停間での増便により，利便性と採算性を向上させることができる．

現在，路線バス事業者は上述した利用動向の把握のために，各利用者がどの出発地 (Origin) のバス停で乗車しどの目的地 (Destination) のバス停で降車したかを示す OD (Origin Destination) を集計した OD データを取得している [2]．OD データは，表1に示すような形式となっている．この表では，行が乗車バス停，列が降車バス停を意味しており，乗車バス停と降車バス停の交差部分がその区間の利用者数を表している（利用者が乗降しないバス停間は空欄としている）．

従来，OD データの取得は，複数の添乗調査員の目視による記録や，利用者へのカード配布・回収といった人手に頼った手法 [3] が主流であった．しかし，複数の添乗調査員が目視によって OD データを記録する手法では，調査員の人件費をかけないために，数か月や1年に1度しか実施できず，限定的な期間の OD データしか得られない．限定的な期間の OD データをもとにしたダイヤの調整では，季節や大規模イベントによる需要増加に対応できないため，満員のために利用者がバスに乗車できない積み残しが発生する可能性がある．他方で，利用者へのカード

配布・回収によってODデータを取得する手法は，添乗調査員が運行中にカードの配布・回収を行うため，カードの配布・回収に時間がかかった場合に運行に支障をきたす．そのため，利用者数の多い混雑した路線バスでは，この手法はODデータの取得に適さない．こうした状況から，人件費等の金銭コストがかからないかつ混雑時でも高い精度でODデータを自動取得する方法が必要となる．

ODデータを自動取得する手法は，ICカード乗車券を用いた手法，カメラを用いた手法，スマートフォンが発するパケットを用いた手法に大別される．ICカードを用いた手法は，ICカードの決済情報とGPS (Global Positioning System) 情報を用いてODデータを取得する [4, 5]．この手法は，混雑時に高い精度でODデータを取得できる一方，ICカードを読み取る機器の導入にかかる金銭コストが高い．カメラを用いた手法は，カメラによりバス車内を撮影した映像を用いてODデータを取得する [6]．この手法は，機材の導入にかかる金銭コストが低い一方，バスの車内が混雑している場合に人物検出の精度が低下するため，混雑時のODデータの取得精度が低下する．スマートフォンが発するパケットを用いた手法は，バス利用者が所持するスマートフォンが発するパケットを用いてODデータを取得する [7, 8, 9, 10]．この手法は，バス利用者のスマートフォンが発するパケットを用いるため，バス車内の混雑状況によらずにODデータを取得できる．また，パケットの収集に用いるパケット収集デバイスは安価であるため，機材の導入にかかる金銭コストも低い．したがって，混雑時のODデータの取得精度，金銭コストの観点から，スマートフォンが発するパケットを用いた手法がODデータを自動取得する手法として適していると考える．詳細は2.1節で議論する．

これまで，スマートフォンが発するパケットを用いたODデータの取得に関す

表 1: ODデータの例

降車バス停 / 乗車バス停	バス停2	バス停3	バス停4
バス停1	5	7	3
バス停2		6	9
バス停3			2

る様々な研究がなされてきた [7, 8, 9, 10, 11]．これらの研究では，各利用者はスマートフォンを 1 台のみ携帯しているという仮定のもと，各スマートフォン固有の値である MAC アドレスの有無を観測することで，その MAC アドレスに対応するスマートフォンを携帯する利用者が乗車中か否かを判断する．具体的には，まず，バス車内に存在する複数の利用者のスマートフォンが発するパケットを常に収集し，それらのパケットに記載される MAC アドレスと，パケットを収集した日時を示すタイムスタンプを記録する．そして，各 MAC アドレスに対応するタイムスタンプから，その MAC アドレスが最初に観測された時間と最後に観測された時間を特定し，これらの時間と合致する出発時刻と到着時刻を持つバス停を推定することで OD データを生成する．例えば，Dunlap らは，利用者のスマートフォンが発する BLE (Bluetooth Low Energy) の広告パケットから MAC アドレスを取得し OD データを取得する手法と，利用者のスマートフォンが発する Wi-Fi のプローブ要求パケットから MAC アドレスを取得し OD データを取得する手法 [8] を提案した．

しかし，従来のスマートフォンが発するパケットを用いた OD データの取得手法を，現在の多くのスマートフォンに対して適用することはできない．プライバシー保護の観点から，iOS 8.0 もしくは Andorid 8.0 以降の OS を搭載したスマートフォンでは，アクセスポイントに接続されるまで，乱数生成したアドレスであるランダム MAC アドレスが使用され，この MAC アドレスは定期的に値が変更されるためである [12, 13]．従来手法では，MAC アドレスが固定されていることを前提として，同じ MAC アドレスが継続して観測されている場合，利用者が乗車中であると推定している．したがって，ランダム MAC アドレスを用いるスマートフォンの場合，従来手法では，定期的な MAC アドレスの変更による元の MAC アドレスの消失を利用者の降車と判断するため，正しい OD データを生成することが困難になる．

ランダム MAC アドレスを用いるスマートフォンが登場して以降，そうしたスマートフォンを追跡する様々な研究がなされてきた．Becker らは，BLE の広告パケットに記載される MAC アドレスと一定時間変更されない値である識別子をペアとして保存しておき，MAC アドレスが変更されたとき，ペアの識別子を基

に新しい MAC アドレスに更新するアドレス・キャリーオーバー・アルゴリズム [14] を提案した．Becker らは，このアルゴリズムを用いることで，最長で 53 分間にわたり iPhone を追跡できたと報告している．しかし，Becker らはこの手法を適用できない場合について，以下を挙げている．

- Android を搭載したスマートフォンの場合，BLE の広告パケットを定期的に送信しないため，アドレス・キャリーオーバー・アルゴリズムの適用が非常に困難である．
- Android もしくは iOS を搭載したスマートフォンの場合，識別子が MAC アドレスと同期して変化することが多く，その場合，アドレス・キャリーオーバー・アルゴリズムの適用が困難である．

これらの問題に対して, 新型コロナウイルス接触確認アプリである BLE の広告パケットを定期的に送信する COCOA (Covid-19 Contact-Confirming Application) を活用することで解決可能であるかを検証した．COCOA の技術的仕様を検証した結果，COCOA は，他者に接触を通知するために BLE の広告パケットを定期的に送信するという特徴を有することから，1 つ目の問題を解決できるこが明らかになった．また，COCOA が送信する BLE の広告パケットが，MAC アドレスと非同期に変化するかつ識別子として使用可能な値を含むかについて検証を行った．検証の結果，MAC アドレスと非同期に変更されるという特徴を有する RSSI (Radio Signal Strength Indicator) を識別子として利用することで，2 つ目の問題を解決できる可能性を見出した．

しかし，路線バスの車内を想定した場合，RSSI を用いたアドレス・キャリーオーバー・アルゴリズムは，以下の要因によりスマートフォンの追跡精度が低下する可能性がある．

- 隣接した座席に利用者が座っている場合，RSSI が同一値となる可能性が高くなり，その場合，アドレスが変更された際に RSSI を用いて利用者を区別できない．
- 識別子となる RSSI の 分解能が整数と低く，識別子間の衝突が起こる可能性が高い．

上記の要因に対し，複数のスマートフォンのうち同時にMACアドレスが変更されたスマートフォンのみをアドレス・キャリーオーバーの対象とすることで，アドレス・キャリーオーバーにRSSIを用いることによるスマートフォンの追跡精度の低下を抑制することが可能であると考えた．従来の手法では車内全てのMACアドレスをアドレス・キャリーオーバーの対象としていたが，路線バスの車内に存在する全てのスマートフォンのMACアドレスの変更が同時に発生する可能性は低いため，アドレス・キャリーオーバーの対象となるMACアドレスの数を減らすことが可能である．アドレス・キャリーオーバーの対象となるMACアドレスの数を減らすことができれば，RSSIが同一値となる可能性や識別子間の衝突の可能性を低くすることができる．

本研究では，複数のスマートフォンにおけるMACアドレスの変更タイミングの違い，すなわちMACアドレスランダム化の非同期性に着目した上記のアプローチを基に，RSSIを識別子としたアドレス・キャリーオーバー・アルゴリズムによるODデータの自動取得を目的とする．

本稿の構成は以下の通りである．2章では，ODデータの自動取得に関する関連研究についてまとめ，関連研究の問題点を述べる．3章では，本提案で用いる新型コロナウイルス接触確認アプリの技術的仕様について述べた後，新型コロナウイルス接触確認アプリが送信するBLEの広告パケットが識別子として使用できる情報を含むかについて検証する．4章では，検証実験の結果から，路線バスの車内環境を考慮したスマートフォンの追跡手法を提案した後，本提案を用いたODデータ自動取得システムの実装について説明する．5章では，提案システムを用いたODデータの取得精度を評価する．6章では，評価結果を元にODデータの取得精度に対する考察を行い，今後の展望について述べる．7章では本研究のまとめを行う．

2. 関連研究

本章では，まず，OD データ自動取得に関する研究について説明する．次に，OD データ自動取得における問題を解決し得る，MAC アドレスランダム化に対する研究について説明する．最後に，MAC アドレスランダム化に対する手法について考察する．

2.1 OD データ自動取得の関連研究

本節では, OD データ自動取得に関する研究として, IC カードを用いた OD データ自動取得手法，カメラを用いた OD データ自動取得手法，スマートフォンが発するパケットを用いた OD データ自動取得手法について説明する．

2.1.1 IC カードを用いた OD データ自動取得手法

Wang らは，IC カードの決済情報と AVL (Automatic Vehicle Location) を用いて OD データを取得するために，トリップチェーンを適用する手法 [4] を提案している．トリップチェーンとは，利用者が降車するバス停は次に乗車するバス停に最も近いバス停と仮定する手法である．この手法を用いることで，利用者が降車したバス停が記録されていない IC カードの決済情報から，利用者の降車バス停を推定することができる．しかし，降車したバス停から徒歩で 1，2 駅離れたバス停で乗車する利用者がいた場合，OD データの推定精度が低下する．そのため，降車したバス停から徒歩で 1，2 駅離れたバス停で乗車する利用者がいた場合，OD データの推定精度が低下する．実際に Wang らの研究では，ショッピングセンターに近いバス停で降車した利用者が最寄りでないバス停で乗車する機会が多かったため，この降車バス停を含む路線の OD データの推定精度は 60%という結果となった．

Huang らは，複数の GPS データの集約により得たバスの軌跡を基に，密度ベースのクラスタリング・アルゴリズムを用いて停車したバス停を特定し，特定したバス停を基に生成した時刻表と IC カードの決済情報に含まれる決済日時を照合

することで，利用者が乗車したバス停を推定する手法 [5] を提案している．通常，バスはバス停に近づくと減速しバス停で停車するため，複数の GPS データをまとめた際，バスの軌跡点はバス停付近に集中する．つまり，路線バスの場合，バス停付近のバスの軌跡点の密度は高くなるという特性がある．同氏らの手法は，この特性を基に，密度ベースのクラスタリングアルゴリズムを用いて，実際にバスが停車したバス停を特定し，特定したバス停を基に生成した時刻表と IC カードの決済情報に記録される決済日時を照合することで，利用者の乗車バス停を推定する．一方，降車バス停の推定はトリップチェーンの適用により推測される．提案手法の評価は，中国蘇州の路線バスを対象として行われており，OD データの推定精度は 92%以上という結果になった．

2.1.2 カメラを用いた OD データ自動取得手法

Takao らは，バス車内を撮影した映像を用いて，機械学習の物体検出モデルと画像処理により OD データを取得する手法 [6] を提案している．バス車内の撮影に用いるカメラはバス前方の天井に設置されており，カメラの画角は全てのバス座席を撮影できるように調整されている．この手法では，GPS のデータからバスの位置や速度，バス停の位置等の情報を取得し，これらの情報からバスが停車しているか，移動しているかを判別している．バスが移動している場合，この手法は，まず，機械学習の物体検出モデルを用いてバス車内を撮影した映像から利用者を切り出す．次に，画像処理により利用者の特徴点を抽出し，その特徴点情報をバス停フォルダに保存する．最後に，保存済みの特徴点情報と新規の特徴点情報を照合し，一致する特徴点数が一定数以上の場合，同一人物と判定し，そうでない場合は別人物と判定する．バスが停止している場合，各バス停フォルダ間で特徴点を照合し，乗車，乗車中，降車の 3 パターンに分類する．以上の手順により，利用者の乗車バス停と降車バス停を推定している．

2.1.3 スマートフォンが発するパケットを用いた OD データ自動取得手法

Kostakos らは，利用者のスマートフォンが発する BLE の広告パケット（BLE デバイスが自分のデバイス情報を周囲の不特定多数のデバイスに送信するためのパケット）から取得した MAC アドレスを活用して OD データを取得する手法 [7] を提案している．バスの利用者を対象に BLE の利用状況に関するアンケートをとった結果，利用者の 73%がスマートフォンを携帯していたが，Bluetooth を有効化していた利用者は 12%であった．Bluetooth を有効化していた利用者は少なかったが，その利用者について収集した OD データの推定精度は約 80%となっている．したがって，Bluetooth を有効化する利用者が増加すれば，高い精度で OD データ取得ができるという点において，BLE の広告パケットを用いた OD データ自動取得は有望である．

Dunlap らは，GPS データと利用者のスマートフォンが発する BLE の広告パケットから MAC アドレスを取得し OD データを取得する手法と，GPS データと利用者のスマートフォンが発する Wi-Fi のプローブ要求パケットから MAC アドレスを取得し OD データを取得する手法 [8] を提案している．GPS データや広告パケット，プローブ要求パケットの収集はスマートフォン内蔵のセンサを使用しており，このスマートフォンはバスの運転席付近に設置され，バスの電源から継続的に電力が供給されている．利用者はスマートフォンを 1 台のみ所持すると仮定し，デバイス固有のアドレスである MAC アドレスを用いて，乗降バス停で MAC アドレスを照合することで OD データを取得している．その他の研究においても，乗降バス停で MAC アドレスを照合する手法を採用している．この手法では，収集した MAC アドレスに対して，以下の条件に従ってフィルタリングを実施することで，バス周辺に存在する非利用者のスマートフォンが発する広告パケット，および，プローブ要求パケットの MAC アドレスを除去し，OD データの推定精度を向上させた．

- 1 回もしくは，ごく少数しか検出されなかった MAC アドレスを除去する．
- 過去の運行記録から，路線の運行時間外に検出された MAC アドレスを除去する．

- GPS データと比較し，バス停位置から指定した距離以上離れているデータを除去する．

Pu らは，利用者のスマートフォンが発する Wi-Fi のプローブ要求パケットから MAC アドレスを取得している．同氏らの手法は，路線バスと他の車両が並走する場合，バス利用者と他の車両に搭乗する非利用者のプローブ要求パケットの特徴が類似することから，非利用者の MAC アドレスの除去が困難になることを考慮し，検出された MAC アドレスを Fuzzy C-Means (FCM) クラスタリングアルゴリズムで利用者クラスタと非利用者クラスタに分類し，利用者クラスタの MAC アドレス数に基づいて Random Forest (RF) 回帰を用いて OD データを取得する手法 [10] を提案している．検証の結果，ほぼ全てのバス停において，推定値の誤差が無視できる程度となっており，既存のフィルタリング手法より推定精度が向上したことが示されている．

2.1.4 OD データ自動取得の関連研究のまとめ

本節では，以下の要件により，これまでに述べた OD データの自動取得手法を比較し，OD データの自動取得に最も適した手法を判断する．以下に，これらの要件が要求される理由を述べる．

要件 1 バス車内の利用者数によらない

バスの利用者が多い区間と少ない区間を的確に把握した上でダイヤを調整することを OD データ取得は目的としているため，バスの利用者数によらず正確な OD データを取得できることが必要となる．

要件 2 金銭コストが少ない

路線バス事業者の 69%が赤字であるため，できる限り少ない金銭コストであることが必要となる．

これらの要件を基に，OD データの自動取得手法を比較した結果を表 2 に示す．まず，IC カードによる OD データの自動取得手法は，IC カード内に乗車・降車バ

ス停が保存されている場合，正確な OD データを取得可能である．しかし，利用者は IC カードを所持している必要がある．また，自動料金収受システムの新規導入に多大な金銭コストがかかる点が問題点として挙げられる．次に，カメラによる OD データの自動取得手法は，支払い方法によらず OD データを取得可能であり，システム導入時の金銭コストも小さい．しかし，バスの利用者が多い場合に人物検出の精度が低下するため，OD データの取得精度が低下することが問題点として挙げられる．最後に，スマートフォンが発するパケットを用いた OD データの自動取得手法は，バス車内の利用者数によらず OD データを取得可能であり，システム導入時の金銭コストが小さい．しかし，利用者が Wi-Fi や Bluetooth 等のパケットを発するスマートフォンを 1 台だけ所持している必要がある．

表 2: OD データの自動取得手法の比較

	IC カード	カメラ	スマートフォン
要件 1	○	×	○
要件 2	△	○	○

上記の比較結果より，本研究では，スマートフォンが発するパケットを用いた OD データの自動取得手法について着目する．スマートフォンが発するパケットを用いた手法は，要件 1，2 を満たすことから OD データの自動取得に最も適していると考えられるためである．

しかし，現在の多くのスマートフォンを想定した場合，スマートフォンが発するパケットを用いる手法を OD データの自動取得手法として利用することはできない．現在のスマートフォン（iOS 8.0 もしくは Andorid 8.0 以降の OS を搭載したスマートフォン）は，プライバシー保護の観点から，自身がアクセスポイント (AP) に接続されるまでの間，乱数生成される，かつ定期的に値が変更されるアドレスであるランダム MAC アドレスを使用しているためである [12, 13]．従来手法では，MAC アドレスが固定されていることを前提として，同じ MAC アドレスが継続して観測され続ける場合，利用者が乗車中であると推定している．したがって，ランダム MAC アドレスを用いるスマートフォンの場合，従来手法では，定期的な MAC アドレスの変更，すなわち MAC アドレスランダム化による

元の MAC アドレスの消失を利用者の降車と判断するため，正しい OD データを生成することが困難となる．

2.2 ランダム MAC アドレスに対する関連研究

ランダム MAC アドレスを使用するスマートフォンに対応した OD データ自動取得手法を確立するため，ランダム MAC アドレスに対する関連研究について調査した．

2.2.1 ランダム MAC アドレスに対する手法

Becker らは，BLE の広告パケットから MAC アドレスと識別子となる一定時間変更されないビット列を抽出し，MAC アドレスが変更されたとき，識別子を基に新しいアドレスに更新するアドレス・キャリーオーバー・アルゴリズムを [14] を提案している．このアルゴリズムは，前処理フェーズと追跡フェーズで構成される．以降で，各フェーズについて説明する．

前処理フェーズ　このフェーズでは，BLE の広告パケットから一定時間変更されないビット列である識別子を選定する．同氏らは，BLE の広告パケットが永続的にデバイス固有となるビット列を含まないと仮定し，一定期間にわたってデバイス固有となるビット列を識別子と定義している．また，同氏らは，識別子間の衝突（異なるデバイスが同一のビット列を識別子として使用すること）を回避するため，ビット長の大きい識別子が望ましいと述べている．

以下のようにして，識別子のビット長の下限を定める．MAC アドレスを追跡する 1 台の BLE デバイスと最大 m 個のその他全ての BLE デバイスを含む領域で，1 時間，対象の BLE デバイスを追跡すると仮定する．また，各デバイスが 15 分に 1 回以下の割合で識別子の候補となるビット列を変更すると仮定する．つまり，対象のデバイスは，1 時間に最大で 4 個のビット列を生成する（各ビット列は一様にランダムに生成されると仮定する）．別の 1 台の BLE デバイスが，これら 4 個のビット列の 1 つと異なるビット列を生成する確率は，少なくとも $(1-2^{-n})^4$

であるため，他の m 個の BLE デバイスが全て対象となるデバイスと異なるビット列を生成する確率は，少なくとも $(1-2^{-n})^{4m} \geqq 1-4m\cdot 2^{-n}$ である（後者の不等式はベルヌーイの不等式を適用して得られる）．したがって，$n \geqq \log_2(16m/p)$ により，p よりも小さい衝突確率（異なるデバイスが同じ値のビット列を識別子として使用する確率）を確保することができる．例えば，$m = 50$ 台，$p = 10^{-3}$ の場合，$n \geqq 19.6$ ビット，すなわち，識別子のビット長の下限は 20 ビットとなる．

追跡フェーズ　このフェーズは, 前処理フェーズで選定した識別子を使用し, MAC アドレスの変化を追跡する．このフェーズの目的は，MAC アドレスを変更した 1 台のデバイスに対し，BLE の広告パケットから抽出した識別子を用いて，MAC アドレスをランダム化する時点を超えてデバイスを追跡することである．

このフェーズでは，hcidump[1]等のパケット収集ソフトウェアにより収集した全ての広告パケットを時系列順に整列し，そのパケットのリストに対し，逐次的に処理を行う．このフェーズで用いるアルゴリズムの処理内容を以下に示す．このアルゴリズムは，対象デバイスの MAC アドレスと識別子リストのペア（標的ペア），現在の MAC アドレスと識別子リストのペア（現在ペア）を用いる．また，標的ペアと現在ペアは MAC アドレスと識別子リストのタプルとし，識別子は複数個存在する場合を考慮してリストとする．

1. 標的ペアを追跡対象となる 1 台のデバイス D の現時点の（ランダム）MAC アドレスと識別子で初期化する．
2. 広告パケットが少なくとも 1 つの識別子を含むか確認する．含む場合は処理 3 に進み，含まない場合は処理 7 に進む．
3. その広告パケットの MAC アドレスと識別子を現在ペアに代入する．
4. 現在ペアの MAC アドレスが標的ペアの MAC アドレスと一致するか確認する．一致する場合は処理 5 に進み，一致しない場合は処理 6 に進む．
5. 現在ペアの識別子リストが標的ペアの識別子リストと一致しない場合，標的ペアの識別子リストを現在ペアの識別子リストに更新する．

[1]http://www.bluez.org/

6. 現在ペアの識別子が標的ペアの識別子リストに含まれている場合，標的ペアの MAC アドレスを現在ペアの MAC アドレスに更新する．また，標的ペアの識別子リストを現在ペアの識別子リストに更新する．

7. 次のパケットに更新し，処理 2 に戻る．

このように，識別子を用いて，MAC アドレスが変更される度に新しい MAC アドレスに更新することによって，MAC アドレスをランダム化する時点を超えてデバイスを追跡することができる．

Becker らは，このアルゴリズムを用いることで，最長で 53 分間にわたり iPhone を追跡できたと報告している．しかし，Becker らは，アドレス・キャリーオーバー・アルゴリズムを適用できない場合について，以下の要因を挙げている．

1. Android を搭載したスマートフォンの場合，BLE の広告パケットを定期的に送信しないため，アドレス・キャリーオーバー・アルゴリズムの適用が非常に困難である．

2. Android もしくは iOS を搭載したスマートフォンの場合，識別子が MAC アドレスと同期して変化することが多く，その場合，アドレス・キャリーオーバー・アルゴリズムの適用が困難である．

2.3 ランダム MAC アドレスに対する手法の考察

2.2.1 項で述べた問題に対するアプローチについて述べる．1 つ目の問題に対して，新型コロナウイルス接触確認アプリである COCOA (Covid-19 Contact-Confirming Application) の技術的仕様から，この問題は COCOA を用いることで解決できると考えた．COCOA は，COCOA をスマートフォン (Andorid, iOS) にインストールした利用者同士が接触した（近距離にいた）ときに，接触相手の利用者情報を記録し，接触した人の中にコロナウイルスに感染したと報告した利用者がいた場合，コロナウイルス感染者と接触があったことを通知するアプリケーションである．COCOA の技術的仕様を調査した結果，COCOA は，他者に接触を通知するために BLE の広告パケットを定期的に送信することが明らかとなっ

アルゴリズム 1 アドレス・キャリーオーバー・アルゴリズム

Input:

- $Packets$: 時系列順に整列したパケットのリスト

Variable:

- $target\_pair = (addr_1, [id_1, id_2, \ldots, id_n])$
  - 対象デバイス D の MAC アドレス $(addr)$ と識別子 (id) リストのペア
- $current\_pair = (addr, [id_1, id_2, \ldots, id_n])$
  - 現在の MAC アドレスと識別子リストのペア

Initialize:

- $target\_pair \leftarrow$ 対象デバイス D の MAC アドレスと識別子

for $p \leftarrow Packets$ **do**
  if p が少なくとも 1 つの識別子を含んでいる **then**
    $current\_pair \leftarrow p$ のアドレスと識別子
    if $current\_pair$ のアドレス $=$ $target\_pair$ のアドレス **then**
      if $current\_pair$ の識別子リスト $\neq$ $target\_pair$ の識別子リスト **then**
        $target\_pair$ の識別子リスト $\leftarrow$ $current\_pair$ の識別子リスト
    else
      if $current\_pair$ の識別子 $\in$ $target\_pair$ の識別子リスト **then**
        $target\_pair$ のアドレス $\leftarrow$ $current\_pair$ のアドレス
        $target\_pair$ の識別子リスト $\leftarrow$ $current\_pair$ の識別子リスト

Result:対象デバイス D の追跡に成功し，新しいアドレスは $current\_pair$ のアドレスとなる

た．2つ目の問題に対しては，COCOA が送信する BLE の広告パケットが，MAC アドレスと非同期に変化するかつ識別子として使用可能な値を含むかが不明であるため，検証する必要がある．3 章では，COCOA の技術的仕様の詳細と検証実験について述べる．

3. COCOA に関する調査と検証

この章では，まず，COCOA の技術的仕様について説明し，COCOA が送信する BLE 広告パケットから，アドレス・キャリーオーバー・アルゴリズムの識別子として有効に働く可能性がある識別子の候補を検討する．次に，上記の検討で明らかとなった識別子の候補が，実際に識別子として使用できるかを検証する．

3.1 COCOA の技術的仕様について

COCOA は，Google と Apple が開発した機能である Exposure Notification を使用することで接触検知を行っている．Exposure Notification は，接触検知を行うために BLE の広告および BLE のスキャンを用いている．

まず，BLE の広告について説明する．BLE の広告とは，BLE デバイスが自分のデバイス情報を，周囲に存在する不特定多数の BLE デバイスに知らせるために，専用のチャネルを通じてパケットをブロードキャストすることである．BLE は，2402 MHz から 2480 MHz まで 2 MHz 刻みで 40 の物理チャネルで動作している．これらのチャネルのうち，2402 MHz，2426 MHz，2480 MHz の 3 つが広告に使用される広告チャネルである．

次に，BLE のスキャンについて説明する．BLE のスキャンとは，BLE デバイスが自分の周囲に存在する BLE デバイスの情報を取得することである．スキャンの種類を以下に示す．COCOA では，パッシブスキャンが使用されている．

パッシブスキャン

広告パケットを受信可能な状態にして，周囲に存在する他の BLE デバイスが広告する広告パケットを取得する．

アクティブスキャン

BLE では，広告パケットに含まれる情報は 31 Byte しかない．そのため，広告パケットの受信後，広告している BLE デバイスに対して，スキャン要求を送信することで追加情報を取得する．

広告とスキャンは, 広告パケットを用いて行われる．広告パケットのフォーマットと内容について説明する．広告パケットのフォーマットを図 1 に示す．広告パケットは，プリアンブル，アクセスアドレス (AA) ，PDU (Protocol Data Unit) ヘッダ，PDU ペイロード，CRC (Cyclic Redundancy Check) フィールドから構成される．

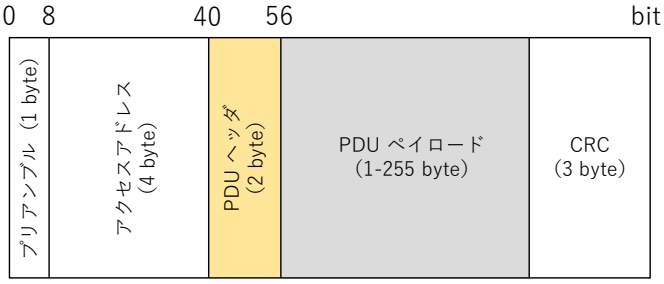

図 1: BLE の広告パケットのフォーマット

各用語について以下に説明する．

プリアンブル

プリアンブルは，信号の読み出しタイミングの同期に用いられるフィールドである．0 と 1 を交互に繰り返す 10101010 もしくは 01010101 が使用される．

アクセスアドレス

アクセスアドレスは, 受信する BLE デバイスが待っているパケットの判別と，受信デバイスでバイト単位のタイミングを取るために用いられるフィールドである．広告パケットの場合, ビット表記で 10001110100010011011111011010110（16 進数では，0x8E89BED6）が使用される．

PDU ヘッダ

図 2 に示すように，PDU ヘッダは，広告やスキャン要求といったタイプを示す PDU タイプ，広告（送信）を示す TxAdd，スキャン（受信）を示す RxAdd，PDU の長さを示す Length 等で構成されるフィールドである．

PDU ペイロード

PDU ペイロードは, 広告パケットが運ぶデータ（ペイロード）を含むフィールドである．このフィールドの用途として，先述した広告やスキャン要求，スキャン要求に対する応答であるスキャン応答，特定のデバイスへの接続を要求する接続要求がある．PDU ペイロードは，図 3，図 4 に示すように，特定のデバイスに対する広告の PDU ペイロードと，不特定多数のデバイスに対する広告の PDU ペイロードでフィールドの構造が異なる．

CRC

CRC は, PDU の誤り検出を行うための誤り検出符号である．2 および 4 ビットまたは奇数個のエラービットがあるとき，誤りを検出できる．

特定のデバイスに対する広告の PDU ペイロードは，図 3 に示すように，広告アドレス (AdvA) およびターゲットアドレス (TargetA) を含む．一致する Target Address を持つデバイスのみが，接続要求を送信することで接続を開始できる．不特定多数のデバイスに対する広告では，PDU を受信したデバイスは，スキャン要求（利用可能な機能に関する情報を要求）または接続要求で応答することができる．このタイプの広告の PDU ペイロードは，図 4 に示すように，Advertising Address (AdvA) と Advertising Data (AdvData) を含むフォーマットで，1 つまたは複数の AD Structure (図 5 参照) を含む可能性がある．最後に，AD タイプは，各 AD structure に含まれる AD Data の内容を定義する（図 6）.

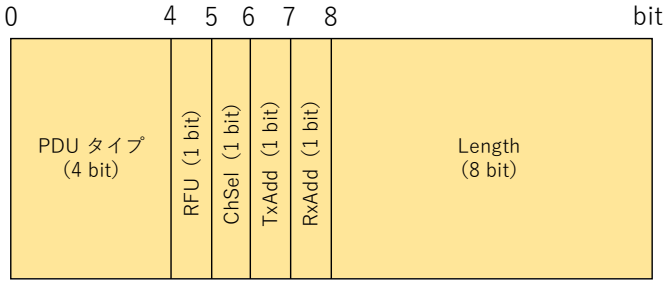

図 2: PDU ヘッダのフォーマット

続いて，Exposure Notification について説明する．Exposure Notification とは，コロナウイルスの拡散を防ぐために，過去 14 日以内に接触した人がその後ウイル

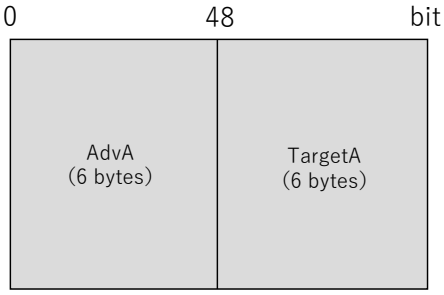

図 3: 特定のデバイスに対する広告の PDU ペイロードのフォーマット

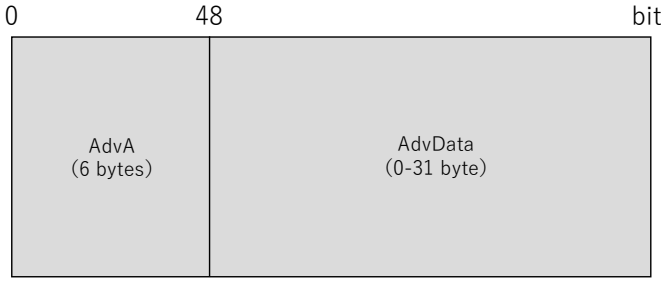

図 4: 不特定多数のデバイスに対する広告の PDU ペイロードのフォーマット

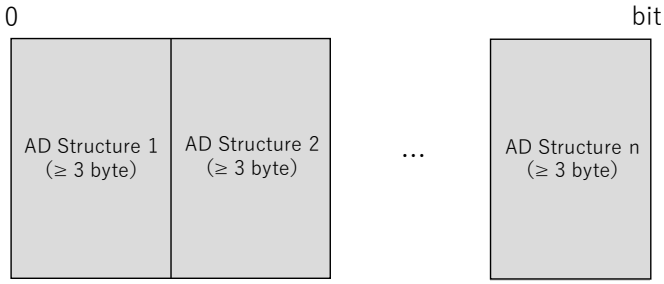

図 5: AdvData のフォーマット

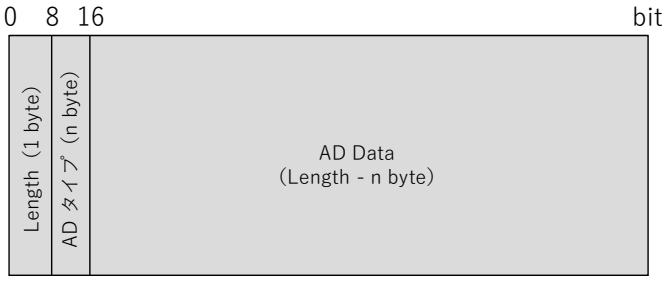

図 6: AD Structure のフォーマット

スに感染していると診断された場合，その可能性について参加者に警告を発する通知のことである [15]．Exposure Notification に関する各用語を以下に説明する．

Exposure Notification Service

BLE デバイスと周囲に存在する BLE デバイスの距離が近いか検出するサービスである．Exposure Notification Service のペイロードを図 7 に示す．Complete 16-bit Service UUID の Service UUID フィールドと Service Data - 16bit UUID フィールドには，Exposure Notification Service であることを示す 16 ビットの UUID 0xFD6F の値が格納されている．このサービスは，接触を検出するために，デバイス間の Rolling Proximity Identifier の近接センシングを可能にするように設計されている．デバイスは，16 ビットの Service UUID を用いて，Exposure Notification Service を広告およびスキャンする．この Service UUID を持つサービスデータタイプには，Rolling Proximity Identifier と関連する暗号化されたメタデータが含まれ，共に定期的に変更される．

Temporary Exposure Key

Temporary Exposure Key は，プライバシーに配慮して 24 時間ごとに生成されるキーである．

Diagnosis Key

Diagnosis Key は，デバイスの所有者がコロナウイルス陽性と診断された際にアップロードされる Temporary Exposure Key のサブセットである．

Rolling Proximity Identifier

Rolling Proximity Identifier は，Temporary Exposure Key から生成され，Bluetooth によりブロードキャストされるプライバシー保護のための識別子である．この識別子は約 15 分ごとに変更され，デバイスの追跡を防ぐ．

AEM (Associated Encrypted Metadata)

AEM は，プロトコルのバージョンや，距離の近似性を高めるための送信電力を伝えるために使用される値である．この値は，無線によるデバイスの

追跡を防ぐために，Rolling Proximity Identifier と同じ周期で，約 15 分ごとに変更される．

Flags			Complete 16-bit Service UUID			Service Data – 16 bit UUID				
Length	タイプ	Flags	Length	タイプ	Service UUID	Length	タイプ	Service Data		
0x02	**0x01**	**0x1A**	**0x03**	**0x03** (Complete 16-bit Service UUID)	**0xFD6F** (Exposure Notification Service)	**0x17**	**0x16** (Complete 16-bit Service UUID)	**0xFD6F** (Exposure Notification Service)	16byte **Rolling Proximity Identifier**	4byte **AEM**

図 7: Exposure Notification のペイロード

最後に，Exposure Notification における広告とスキャンの手順をそれぞれ以下に示す．

広告の手順

1. Temporary Exposure Key が生成される．
2. Temporary Exposure Key から Rolling Proximity Identifier が生成される．
3. Exposure Notification Service，Rolling Proximity，AEM のデータを広告パケットに挿入し，ブロードキャストする．ブロードキャスト間隔は，200〜270 ミリ秒とすることが推奨されている．
4. 約 15 分（10 分以上 20 分未満のランダムな時間）経過後，2 に戻る．
5. 24 時間経過後，1 に戻る．

スキャンの手順

1. 周囲に広告パケットを送信している Bluetooth デバイスがないかスキャンする．
2. 受信した広告パケットが，Exposure Notification Service であった場合，収集した日時を示すタイムスタンプ，BLE デバイスが受信した電波強度を示

す RSSI (Radio Signal Strength Indicator)，Rolling Proximity Identifier，AEM を保存する．

3. 5 分以内に 1 に戻る．

3.1.1 COCOA の技術的仕様に対する考察

2.2.1 項で述べたように，アドレス・キャリーオーバー・アルゴリズムを適用するためには，識別子として，一定期間にわたってデバイス固有となるビット列を選定する必要がある．

この点に関して，Rolling Proximity Identifier は約 15 分間固定され，COCOA を使用するスマートフォンで固有であることから，識別子として有効に働く可能性がある．また，路線バスの車内環境を想定した場合，基本的に路線バス利用者は乗車中に動かないことから，RSSI は揺らぐがばらつきが小さくなると考えられるため，識別子として有効に働く可能性がある．

以上の考えから，Rolling Proximity Identifier と RSSI を識別子として使用可能なビット列の候補として選定する．次節では，これらの候補が実際に識別子として使用できるかについて検証する．

3.2 検証実験

本節では，2.2.1 節で述べた 2 つ目の問題（現在のスマートフォンの MAC アドレスは識別子の値の変化に同期してランダムに変更されるため，アドレス・キャリーオーバーが困難であるという問題）を解決可能な識別子を選定することを目的として，前節で候補として選定した情報が識別子としてアドレス・キャリーオーバーに利用可能であるかを検証する．具体的には，実際に COCOA のパケットを収集する実験を行い，収集した広告パケットの Exposure Notification に含まれる Rolling Proximity Identifier と RSSI を識別子として使用できるかについて検証する．

3.2.1 実験方法

本実験は，Bluetooth 通信モジュールを収集デバイスに取り付け，BLE の広告パケットを収集するソフトウェアを用いて，広告パケットを収集した．なお，収集した広告パケットは pcap 形式で保存した．

本実験の条件を表 3 に示す．今回対象としたデバイスは Google 社製 Pixel 5a と Apple 社製 iPhone 11 とした．これらのデバイスを対象とした理由として，それぞれ搭載している OS（Android と iOS）が現在のスマートフォンで使用されている代表的な OS であるためである．また，OS のバージョンは Android 11 と iOS 14.8 であり，デバイスの状態として Bluetooth を ON にした．

広告パケットの収集時間は 30 分とした．本実験では，MAC アドレスと Rolling Proximity Identifier の変更タイミングの観測を目的としているため，この 30 分という時間設定は，Rolling Proximity Identifier が変更するまでに約 15 分間かかることを考慮した設定となっている．

表 3: 実験条件

	Pixel 5a	iPhone 11
OS	Android 11	iOS 14.8
デバイスの状態	Bluetooth ON	
収集時間	30 分	

3.2.2 実験環境

実験は研究室内で行った．図 8 に示すように，検証対象のスマートフォン以外の電波を遮蔽するため，電波暗箱に検証対象のスマートフォン 1 台と BLE 広告パケット収集デバイス 1 台を入れた．

BLE 広告パケット収集デバイスとして，Bluetooth 通信モジュール TP-Link UB400 を取り付けた Raspberry Pi 3 Model B+を使用した．Raspberry Pi 3 Model B+は Bluetooth 通信モジュールを内蔵しているが，時折，動作が不安定になることから外付けの Bluetooth 通信モジュールを使用している．広告パケットの収

集には，BLE の広告パケットの収集に特化したパケット収集ソフトウェアである hcidump を用いた．

図 8: 検証実験環境

3.2.3 実験結果

まず，Pixel 5a を対象とした場合の検証結果について述べる．MAC アドレスと Rolling Proximity Identifier の時系列変化を表 4 に示す（この表の時間は，MAC アドレスが変更された時間を示している）．表 4 より，05:07:45〜05:14:50 と 05:19:10〜05:27:44 のとき，Rolling Proximity Identifier が変化していないことがわかる．つまり，これらの期間は MAC アドレスと Rolling Proximity Identifier の変更タイミングが非同期であると言える．続いて，各 MAC アドレスにおける RSSI の時系列変化を図 9 に示す．図 9 より，RSSI はおよそ −31 dB〜−37 dB の範囲に収束していることから，Pixel 5a の RSSI は一定期間にわたってある範囲内に値が固定されていることがわかる．

表 4: MAC アドレスと Rolling Proximity Identifier の時系列変化 (Pixel 5a)

時間	MAC アドレス	Rolling Proximity Identifier
05:01:29	79:15:a8:3f:33:7b	c3f68470cb. . .80f420aac6
05:07:45	47:3b:43:fe:94:f7	bf0b33f909. . .1b72e71042
05:14:50	5a:f3:5d:c7:e8:e1	bf0b33f909. . .1b72e71042
05:19:10	69:c1:a9:1e:1c:70	8a5e602b50. . .4107d51997
05:27:44	68:b9:b0:15:0d:ae	8a5e602b50. . .4107d51997
05:30:31	72:46:f2:5b:4f:42	83dbb8234b. . .7fdd27ab4c

次に，iPhone 11 を対象とした場合の検証結果について述べる．MAC アドレスと Rolling Proximity Identifier の時系列変化を表 5 に示す．表 5 より，すべての時間（04:03:20，04:10:44，04:23:52）において，MAC アドレスと Rolling Proximity Identifier の変更タイミングが同期していることがわかる．続いて，各 MAC アドレスにおける RSSI の時系列変化を図 10 に示す．図 10 より，RSSI は −7 dB〜−15 dB の範囲で外れ値をとっている一方，多くの RSSI は −31 dB〜−37 dB の範囲に収束している．したがって，iPhone 11 の RSSI も一定期間にわたってある範囲内に値が固定されていることがわかる．

表 5: MAC アドレスと Rolling Proximity Identifier の時系列変化 (iPhone 11)

時間	MAC アドレス	Rolling Proximity Identifier
04:03:20	07:69:0c:24:73:43	8f9a557dbb. . .fa74949f0e
04:10:44	2e:ce:8d:80:5d:11	978af6db5b. . .ac9a0f3008
04:23:52	5a:f3:5d:c7:e8:e1	de68b25945. . .69dfa3ea22

3.2.4 考察

3.2.3 節の結果より，多くの場合，Rolling Proximity Identifier と RSSI は MAC アドレスと非同期に変化しない一方，RSSI は一定期間にわたってある範囲内に値が固定されることが明らかとなった．このことから，RSSI を識別子として用い

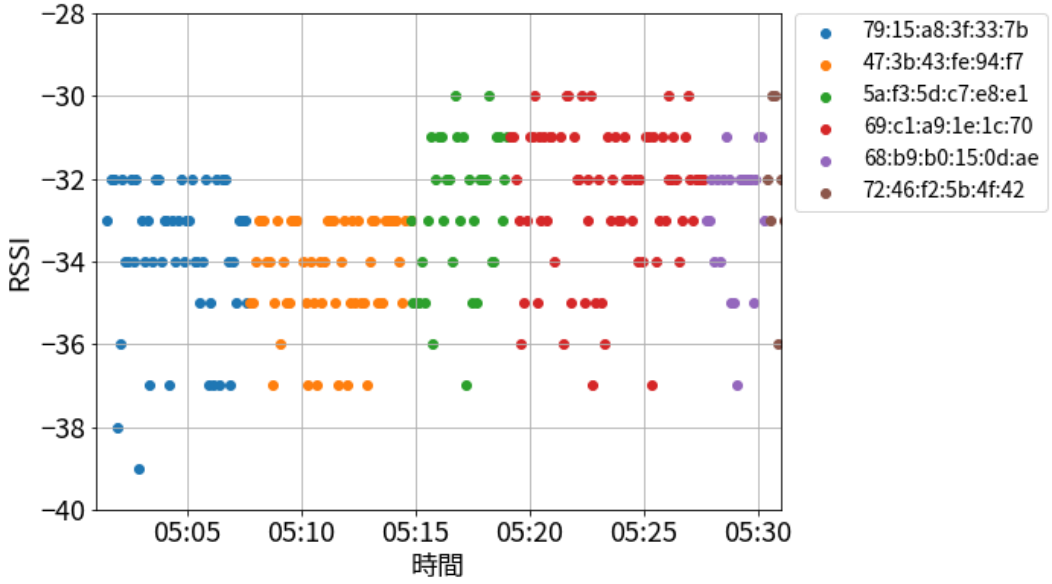

図 9: 各 MAC アドレスにおける RSSI の時系列変化 (Pixel 5a)

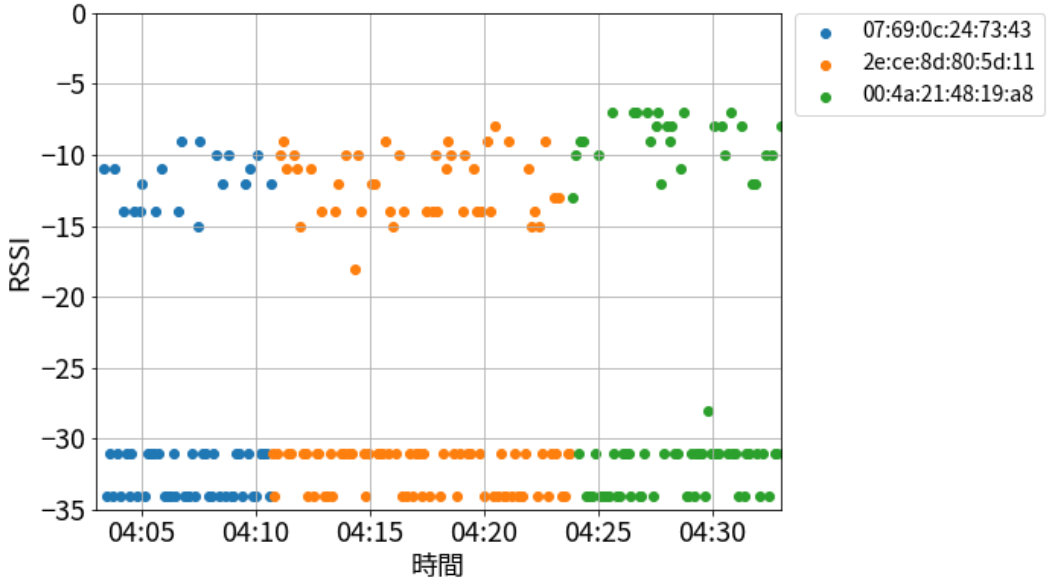

図 10: 各 MAC アドレスにおける RSSI の時系列変化 (iPhone 11)

たアドレス・キャリーオーバーにおいて RSSI のばらつきを許容することで，スマートフォンを追跡できると考えた（図 11 参照）．

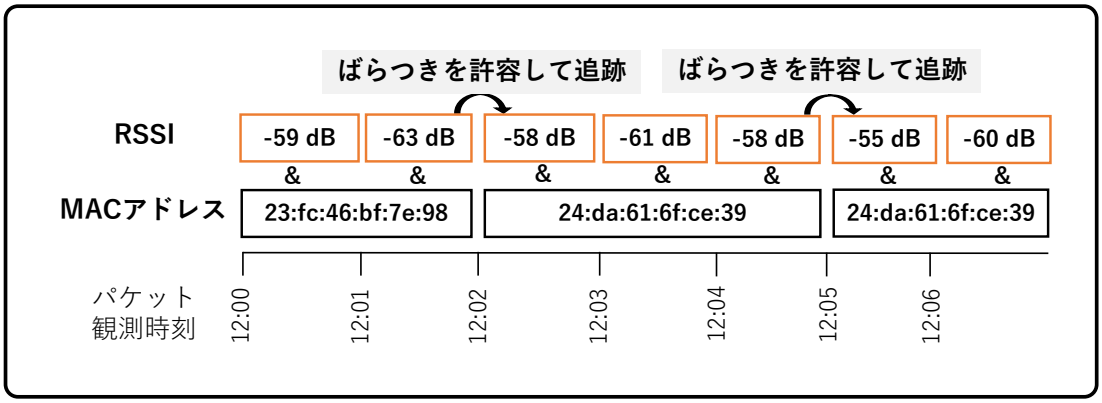

図 11: RSSI を用いたアドレス・キャリーオーバー

しかし，RSSI を識別子として使用した場合，RSSI のばらつき以外にも以下の要因によりスマートフォンの追跡精度が低下する可能性がある．

- 隣接した座席に利用者が座っている場合，RSSI が同一値となる可能性が高くなり，その場合，各デバイスにおいて固有の値でなくなる．
- ビット長が短く，識別子間の衝突が起こる可能性が高い．

上記の要因に対し，複数のスマートフォンにおける MAC アドレスの変更タイミングの違い，すなわち，MAC アドレスランダム化の非同期性に着目することで，スマートフォンの追跡精度の低下を抑制できると考えた．路線バスの車内において，車内に存在する全てのスマートフォンのランダム MAC アドレスが同時に変更される可能性は極めて低いためである．この考えを基に，複数のスマートフォンのうち，MAC アドレスが同時に変更されたスマートフォンのみをアドレス・キャリーオーバーの対象とすることで，スマートフォンの追跡精度の低下を抑制する．このアプローチにより，上記の要因によるスマートフォンの追跡精度の低下を抑制できる理由を以下に述べる．

1 つ目の要因について，MAC アドレスが変更されて一定時間内に観測された新しい MAC アドレスのうち，隣接した座席に座る利用者の MAC アドレスが含ま

れる可能性は低いと考えられるためである．上述したように，MAC アドレスの変更間隔は約 600 秒であるため，路線バスの 1 回の運行にかかる所要時間を 1 時間と仮定すると，1 時間で 6 回のアドレス変更が行われる．この 6 回のアドレス変更において，隣接した座席に座る利用者の MAC アドレスが，MAC アドレスが変更されて一定時間内に観測された新しい MAC アドレスに含まれる可能性は低いと考えられる．

2 つ目の要因について，MAC アドレスが変更されて一定時間内に観測された新しい MAC アドレスは，バスの利用者に対して少ないため，識別子間の衝突が起こる確率は低いと考えられるためである．例えば，バスの利用者の 80 名のうち，新しく観測された MAC アドレスを持つスマートフォンを所持する利用者を，利用者全体の 10%である 8 名と仮定し，識別子となる RSSI のビット長を 8 bit とする．その場合，2.2.1 項の計算式から RSSI の衝突確率は 12.5%となるため，衝突確率が低いことがわかる．

次章では，路線バスの車内環境を考慮した上記のアプローチを基に，複数スマートフォンの MAC アドレスランダム化の非同期性を用いた OD データ自動取得手法を提案する．

4. 複数スマートフォンによるMACアドレスランダム化の非同期性を用いたODデータ自動取得手法

本章では，複数スマートフォンによるMACアドレスランダム化の非同期性を用いたODデータ自動取得手法を提案する．まず，提案手法の概要について述べ，次に，路線バスの車内環境を想定したODデータ自動取得システムについて述べる．

4.1 提案手法の概要

本研究では，複数スマートフォンによるランダムMACアドレスの更新の非同期性を用いたODデータ自動取得手法を提案する．この提案手法では，RSSIを用いたアドレス・キャリーオーバーにより，2.2.1項で述べたMACアドレスと識別子が同期的に変わる問題を解決する．3.2節の検証実験において，RSSIはばらつきがある一方，一定期間にわたってある範囲内に値が固定されると明らかになったため，RSSIのばらつきを許容してアドレス・キャリーオーバーすることでスマートフォンを追跡する．また，3.2.4項で述べたように，RSSIを識別子として用いるだけではスマートフォンの追跡精度が低下するため，アドレス・キャリーオーバーの対象となるMACアドレスを制限することで，スマートフォンの追跡精度を向上を通して，ODデータ取得精度を向上させる．

さらに，本提案手法は，バスの非利用者の除去や利用者の乗降の判別といったODデータ取得精度向上のための工夫を取り入れている．バスで収集する広告パケットには，バスの利用者の他に，バス周辺の歩行者やバスと並走する車両の搭乗者といったバスの非利用者のスマートフォンが発する広告パケットが含まれるため，広告パケットに含まれるMACアドレスのうち，一定時間観測されたMACアドレスをバスの利用者とする工夫を取り入れている．また，バスの利用者の乗車によるMACアドレスの出現と降車によるMACアドレスの消失が想定されるため，（バスの非利用者の除去と同様に）一定時間観測されたMACアドレスを乗車中の利用者とし，一定時間観測されなかったMACアドレスを降車したバスの利用者とする工夫を取り入れている．さらに，MACアドレスの消失に関して，

MACアドレスの変更によるMACアドレスの消失と利用者の降車によるMACアドレスの消失を判別するため，変更後のMACアドレスの候補のうち，変更前のMACアドレスと変更後のMACアドレスのRSSIの差が閾値未満かつ最も近いMACアドレスが存在する場合にMACアドレスを変更し，存在しない場合に降車によってMACアドレスが消失したとする工夫を取り入れている．このような手法により，収集したBLEの広告パケットからODデータを取得する．

まず，提案手法を用いる上で設定が必要となるパラメータを以下に述べる．

$rssi\_th$

: アドレスの変更により出現もしくは消失したMACアドレスと，利用者の乗降により出現もしくは消失したMACアドレスを判別するために必要となるRSSIの閾値であり，単位はdBである．

$passenger\_th$

: バスの利用者と非利用者を区別するための閾値であり，単位は秒である．つまり，$passenger_th$秒以上観測されたMACアドレスをバスの利用者とする．

$extract\_int$

: MACアドレスが変更されて一定時間内に観測された新しいMACアドレスを抽出するための時間間隔であり，単位は秒である．

$timetable\_int$

: 利用者観測データから，時刻表の発車時刻と合致する乗車・降車時刻を持つ利用者を判別するための時間間隔であり，単位は秒である．バスが時刻表通りに発着しないことを考慮し，時刻表の発車時刻から$\pm timetable_int$秒以内に観測されたMACアドレスをバスの利用者としている．

次に，提案手法のアルゴリズムをアルゴリズム2に示す．このアルゴリズムは，時系列順に整列したパケットのリスト，観測アドレス辞書，現在アドレス辞書，利用者観測データを用いる．観測アドレス辞書と現在アドレス辞書は，MACアドレスをキー，MACアドレスに関する情報を抽出したアドレス情報を値とする

辞書型のデータである．アドレス情報には，MAC アドレス，変更前の MAC アドレス，観測開始時刻，観測終了時刻，RSSI，広告パケットのペイロード長，利用者の乗車を示す乗車中フラグ，利用者の降車を示す降車フラグが含まれる．また，利用者観測データは 4 桁の数列である利用者 ID をキーに持ち，観測開始時刻と観測終了時刻を値に持つ辞書型のデータである．アルゴリズム 2 の処理内容を以下に述べる．

1. COCOA のパケットかを確認する．

   (a) パケットのペイロードに Exposure Notification (0xFD6F) が含まれているか確認する．含まれている場合は処理 2 に進み，含まれていない場合は処理 4 に進む（アルゴリズム 2 の 2 行目）．

2. COCOA のパケットから必要な情報を抽出する．

   (a) COCOA のパケットから，MAC アドレス，RSSI，ペイロード長，タイムスタンプを抽出し，現在アドレス辞書に MAC アドレス，RSSI，ペイロード長を代入する．また，観測開始時刻と観測終了時刻にはタイムスタンプの値を代入する．また，変更前の MAC アドレスは空，乗車中フラグは False である（アルゴリズム 2 の 3 行目）．

3. 既出の MAC アドレスか確認する．

   (a) 現在アドレス辞書の MAC アドレスが観測アドレス辞書に含まれているか確認する（アルゴリズム 2 の 6 行目）．

   (b) 含まれている場合，観測アドレス辞書のその MAC アドレスに対応する観測終了時刻を現在アドレス辞書の観測終了時刻に更新する．また観測アドレス辞書のその MAC アドレスに対応する RSSI リストに，現在アドレス辞書の RSSI を追加する（アルゴリズム 2 の 9 行目）．

   (c) 含まれていない場合，現在アドレス辞書を観測アドレス辞書に追加する（アルゴリズム 2 の 11 行目）．

4. 次のパケットに更新し，処理 1 に戻る．処理するパケットがなくなった場合に処理 5 に進む．

5. バスの非利用者を除去する（詳細はアルゴリズム 3 参照）．

   (a) 同じ MAC アドレスに対する観測開始時刻と観測終了時刻の差が $passenger\_th$ を超えた場合にバスの利用者とする（アルゴリズム 3 の 7 行目）．

   (b) MAC アドレスに対応する全ての RSSI の平均をその MAC アドレスの RSSI とする（アルゴリズム 3 の 8 行目）．

   (c) 観測アドレス辞書を観測終了時刻の昇順に整列する（アルゴリズム 3 の 14 行目）．

6. MAC アドレスをアドレス・キャリーオーバーするか，降車によって消失したとするかを判別する（詳細はアルゴリズム 4 参照）．

   (a) 観測アドレス辞書の各 MAC アドレスに対応する観測終了時刻から，観測終了時刻に $extract\_int$ 秒足した時間に出現した MAC アドレスをアドレス・キャリーオーバーの対象となる MAC アドレスとして抽出する（アルゴリズム 4 の 10〜11 行目）．

   (b) 消失した MAC アドレスに対応する RSSI と抽出した全ての MAC アドレスの RSSI の差を計算し，最も RSSI の差が小さい MAC アドレスを選択する（アルゴリズム 4 の 13〜15 行目）．

   (c) 対象の MAC アドレスが以下の条件を満たすか確認する（アルゴリズム 4 の 17 行目）．

      - RSSI の差が最も小さく，その RSSI の差が $rssi\_th$ より小さい．
      - 消失した MAC アドレスと対象の MAC アドレスのペイロード長が同じである．

   (d) 条件を満たす場合，対象の MAC アドレスを消失した MAC アドレスの変更後の MAC アドレスとして，アドレス・キャリーオーバーする（アルゴリズム 4 の 18 行目）．

(e) 条件を満たさない場合，MAC アドレスは利用者の降車によって消失したとする（アルゴリズム 4 の 20 行目）.

7. 利用者観測データを生成する（詳細はアルゴリズム 5 参照）.

(a) 観測アドレス辞書において，変更前のアドレスを持たないアドレスをアドレスリストに格納する（アルゴリズム 5 の 4 行目）.

(b) アドレスリストの先頭アドレスをターゲットアドレスとし，キャリーオーバーリストに追加する（アルゴリズム 5 の 7 行目）.

(c) ターゲットアドレスを変更前のアドレスとする情報辞書が観測アドレス辞書に存在するか確認する（アルゴリズム 5 の 12 行目）.

(d) 存在する場合，キャリーオーバーリストに情報辞書のアドレスを追加する（アルゴリズム 5 の 13 行目）．この情報辞書のアドレスをターゲットアドレスとし，処理 (c) に戻る（アルゴリズム 5 の 14 行目）.

(e) 存在しない場合，以下の利用者 ID をキー，観測開始時刻と観測終了時刻を値とする辞書型のデータである時刻辞書（$time\_dict$）を利用者観測データに追加する（アルゴリズム 5 の 18 行目）.

- 利用者 ID（キー）: 利用者に付与される 4 桁の数字.
- 観測開始時刻（値）: 利用者の観測開始時刻．キャリーオーバーリストの先頭アドレスの観測開始時刻を代入する.
- 観測終了時刻（値）: 利用者の観測終了時刻．キャリーオーバーリストの末尾アドレスの観測終了時刻を代入する.

次のアドレスに更新し，処理 (c) に戻る（キャリーオーバーリストは空にする）.

8. OD データを生成する（詳細はアルゴリズム 6 参照）.

(a) 利用者の観測開始時間と観測終了時間をそれぞれ格納する（アルゴリズム 6 の 6〜7 行目）.

(b) 時刻表リストと比較し，観測開始時間に最も近い発車時刻と観測終了時間に最も近い発車時刻を，乗車時刻と降車時刻に代入する（アルゴリズム 6 の 8〜9 行目）.

(c) 時刻表リストの中で, 乗車時刻と降車時刻に発車時刻が該当するバス停を乗車バス停と降車バス停に代入する（アルゴリズム 6 の 10, 12 行目）.

(d) 観測開始時間と乗車時刻の差と観測終了時刻と降車時刻の差が, $timetable\_int$ 以下であるかを確認する（アルゴリズム 6 の 13 行目）.

(e) $timetable\_int$ 以下の場合，OD データにおいて，処理 3 で推定した乗車バス停と降車バス停を持つ OD の数を 1 増やす（アルゴリズム 6 の 14 行目）.

アルゴリズム 2 路線バス用のアドレス・キャリーオーバー・アルゴリズム

Input:

- *Packets*: 時系列順に整列したパケットのリスト

Variable:

- *observed_dct*: 観測アドレス辞書
- *current_dct*: 現在アドレス辞書
- *passenger_data*: 利用者観測データ

Output:

- *od_data*: OD データ

for $p \leftarrow Packets$ **do**
  if p が Exposure Notification を含んでいる **then**
    current_dct ← p の MAC アドレス, RSSI, ペイロード長, タイムスタンプ
    o_addrs ← *observed_dct* のアドレス
    c_addr ← *current_dct* のアドレス
    if *c_addr* ∈ *o_addrs* **then**
      observed_dct[*c_addr*] の観測終了時刻 ← *current_dct*[*c_addr*] の観測終了時刻
      observed_dct[*c_addr*] の RSSI リストに *current_dct*[*c_addr*] の RSSI を追加
      observed_dct[*c_addr*] を {*c_addr*: *observed_dct*[*c_addr*]} で更新
    else
      observed_dct に *current_dct* を追加
noise_eliminate_addrs ← PASSENGER_FLITERING(*observed_dct*)
ADDRESS_CARRYOVER(*observed_dct*, *noise_eliminate_addrs*)
passenger_data ← PASSENGER_DATA_GENERATION(*observed_dct*)
od_data ← OD_DATA_GENERATION(*passenger_data*)

アルゴリズム 3 非利用者を除去する関数

```
1: function PASSENGER_FLITERING(observed_dct)
2:     o_addrs ← observed_dct のアドレス
3:     for o ← o_addrs do
4:         fst ← observed_dct[o] の観測開始時刻
5:         lst ← observed_dct[o] の観測終了時刻
6:         if lst − fst ≥ passenger_th then
7:             observed_dct[c_addr] の乗車中フラグを True にする
8:             observed_dct[c_addr] の RSSI ← observed_dct[c_addr] の RSSI リスト内の全 RSSI
   の平均
9:         else
10:            observed_dct[c_addr] を削除
11:    o_addrs ← observed_dct のアドレス
12:    for o ← o_addrs do
13:        noise_eliminate_data に [o, lst] を追加
14:    noise_eliminate_data を観測終了時刻の昇順に整列
15:    noise_eliminate_addrs ← noise_eliminate_data のアドレス
16: return noise_eliminate_addrs
```

アルゴリズム 4 アドレス・キャリーオーバーを行う関数

```
1: function ADDRESS_CARRYOVER(observed_dct, noise_eliminate_addrs)
2:     for ne ← noise_eliminate_addrs do
3:         o_addrs ← observed_dct のアドレス
4:         lo_time ← observed_dct[ne] の観測終了時刻
5:         up_time ← lo_time + extract_int
6:         for o ← o_addrs do
7:             fst ← observed_dct[o] の観測開始時刻
8:             if lo_time ≤ fst and fst ≤ up_time then
9:                 addrs に o を追加
10:        carryover_dct に {ne: addrs} を追加          ▷ ne がキーで co_addrs が値の辞書
11:        for co_addrs ← carryover_dct[ne] do
12:            rssi_df ← | disappeared_dct[ne] の RSSI − observed_dct[co_addrs] の RSSI|
13:        min_rssi_df ← min rssi_df
14:        rssi_idx ← rssi_df で min_rssi_df と一致する値の添え字
15:        target_addr ← carryover_dct[ne][rssi_idx]
16:        if min_rssi_df < rssi_th and observed_dct[target_addr] のペイロード長 =
   observed_dct[ne] のペイロード長 then
17:            observed_dct[target_addr] の変更前の MAC アドレス ← observed_dct[ne] の MAC
   アドレス
18:        else
19:            observed_dct[ne] の降車フラグを True にする
20:        carryover_dct[ne] を削除
21:        for c_key ← carryover_dct のキー do
22:            if target_addr ∈ carryover_dct[c_key] then
23:                carryover_dct[c_key] から target_addr を削除
```

アルゴリズム 5 利用者観測データを生成する関数

```
1: function PASSENGER_DATA_GENERATION(observed_dct)
2:     o_addrs ← observed_dct のアドレス
3:     for o ← o_addrs do
4:         if observed_dct[o] の変更前アドレスがない then
5:             addr_list に o を追加
6:     for addr ← addr_list do
7:         target_addr = addr
8:         carryover_list = []
9:         flag = True
10:        while flag do
11:            for o ← o_addrs do
12:                if observed_dct[o] の変更前アドレス = target_addr then
13:                    carryover_list に o を追加
14:                    target_addr ← o
15:                    break
16:                else
17:                    time_dict ← {fst : observed_dct[carryover_list[0]] の観測開始時刻, lst :
   observed_dct[carryover_list[−1]] の観測終了時刻 }
18:                    passenger_data に {passenger_id : time_dict} を追加
19:                    flag = False
    return passenger_data
```

アルゴリズム 6 OD データを生成する関数

function OD_DATA_GENERATION(*passenger_data*)

Variable:

- *od_data*: *bus_stop_list* × *bus_stop_list* の 2 次元配列（各要素は 0 で初期化）

1: *bus_stop_list* ← 時系列順に整列したバス停（1 運行分）
2: *time_table_list* ← 時系列順に整列したバス停の発車時刻（1 運行分）
3: *ids* ← *passenger_data* の利用者 ID
4: **for** *id* ← *ids* **do**
5: fst ← *passenger_data*[*id*] の観測開始時刻
6: *lst* ← *passenger_data*[*id*] の観測終了時刻
7: *c_fst* ← fst に最も近い *time_table_list* の発車時刻
8: *c_lst* ← *lst* に最も近い *time_table_list* の発車時刻
9: *origin_idx* ← *time_table_list* で *c_fst* と一致する値の添え字
10: *o_stop* ← *bus_stop_list*[*origin_idx*]
11: *destination_idx* ← *time_table_list* で *l_fst* と一致する値の添え字
12: *d_stop* ← *bus_stop_list*[*destination_idx*]
13: **if** | fst - *c_fst* | ≤ *timetable_int* ∩ | *lst* - *c_lst* | ≤ *timetable_int* **then**
14: *od_data*[*o_stop*][*d_stop*] ← *od_data*[*o_stop*][*d_stop*] + 1
return *od_data*

4.2 路線バスの車内環境を想定したODデータ自動取得システム

路線バスの車内環境を想定して構築したODデータ自動取得システムを構成するフェーズを以下に示す.

1. パケット収集フェーズ

   事前に，路線バスの1運行におけるBLEの広告パケットを収集する.

2. アドレス・キャリーオーバーフェーズ

   前フェーズで収集した広告パケットを入力とし，4.1節で述べた路線バス用のアドレス・キャリーオーバー・アルゴリズムを用いてアドレスのキャリーオーバーを行い，利用者とその利用者の観測開始時刻と観測終了時刻が1セットとなった利用者観測データを得る.

3. ODデータ生成フェーズ

   前フェーズで取得した利用者観測データを入力とし，ある1つの運行で停車するバス停の発車時刻と観測開始時刻と観測終了時刻が合致する利用者観測データをカウントすることで，ODデータを生成する.

各フェーズについて以下に述べる.

4.2.1 パケット収集フェーズ

パケット収集フェーズでは，3.1節で定義したBLEの広告パケットを取得する．BLE広告パケットの収集は，路線バスが搭載するRaspberry Piをベースとしたデジタルタコグラフ車載器 (パインベース社，VR-1000) にBluetooth通信モジュールを取り付け，hcidumpを用いて行う．収集したBLE広告パケットはpcap形式で保存する．図12で示すように，デジタルタコグラフ車載器は運転手席のほぼ真上に取り付けられている.

3.1節で述べたように，COCOAは周囲にBLEデバイスがないかスキャンを行った後に広告パケットの送信を行っており，仕様上，200〜270ミリ秒間隔で広告パケットを送信していると考えられる．しかし，COCOAを使用しているバ

図 12: デジタルタコグラフ車載器の設置箇所

ス利用者が 1 名の場合，スキャンにより周囲に BLE デバイスが存在しないと判断されるため，200〜270 ミリ秒間隔での広告パケットの送信は行われない．この場合，COCOA は Rolling Proximity Identifier が変更されたタイミングでのみ広告パケットを送信するようになる．上記のタイミングでのみ広告パケットが送信される場合，収集される MAC アドレスは毎度異なると考えられるため，変更された MAC アドレスと利用者の乗車により新しく出現した MAC アドレスを判別できない．この問題を解決するため，デジタルタコグラフ車載器は一定時間ごとに Bluetooth の ON/OFF を切り替えるように設定されている．こうすることで，COCOA は新たに BLE デバイスが出現したと錯覚するため，Bluetooth の ON/OFF を切り替えるタイミングで，COCOA が広告パケットを送信するようになる．

なお，このフェーズでは路線バスの 1 運行分の BLE 広告パケットを収集する．

4.2.2 アドレス・キャリーオーバーフェーズ

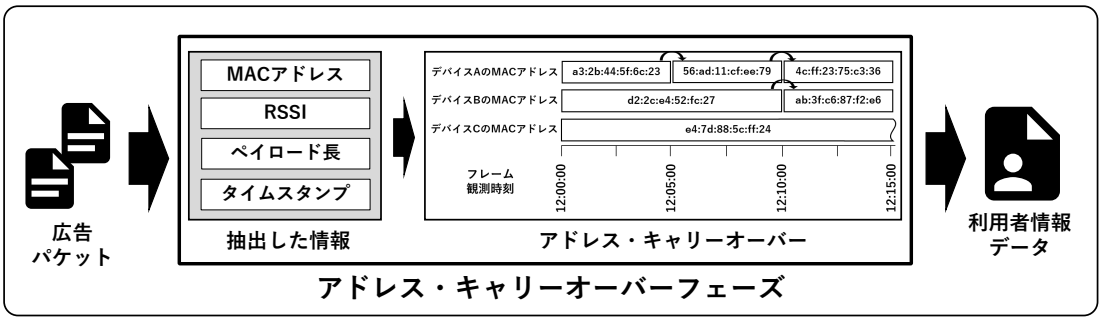

図 13: アドレス・キャリーオーバーフェーズの概要図

アドレス・キャリーオーバーフェーズの概要を図 13 に示す．まず，前フェーズで収集した路線バスの 1 運行分の BLE 広告パケットを入力とし，これらの広告パケットから，MAC アドレス，RSSI，ペイロード長，観測開始時刻と観測終了時刻を決定するために用いるタイムスタンプを抽出する．次に，抽出した MAC アドレス，RSSI，ペイロード長，タイムスタンプを基にアドレス・キャリーオーバーを行う．なお，プライバシー保護の観点から，静的 MAC アドレスが紛れた場合

に備える．MAC アドレスは日毎に変更したソルトを加えたハッシュ関数を用いて，元の MAC アドレスが類推不可能な文字列に変換（匿名化）している．最終的な出力として，表 6 に示すように，4 桁の利用者 ID とその利用者 ID に対応する観測開始時刻，観測終了時刻が 1 セットとなっている利用者観測データが得られる．

表 6: 利用者観測データの例

利用者 ID	観測開始時刻	観測終了時刻
0001	12:00:00	12:15:00
0002	12:00:00	12:15:00
0003	12:10:00	12:20:00
⋮	⋮	⋮

4.2.3 OD データ生成

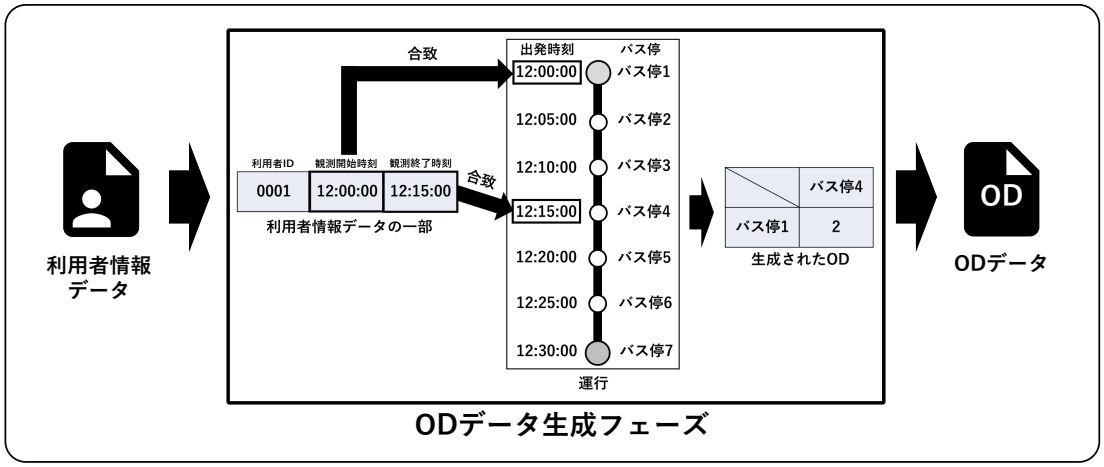

図 14: OD データ生成フェーズの概要図

OD データ生成フェーズの概要を図 14 に示す．このフェーズでは，前フェーズで取得した利用者観測データを入力とし，ある 1 つの運行で停車するバス停の発

車時刻と観測開始時刻と観測終了時刻が合致する利用者観測データをカウントすることで，OD データを生成する．

具体例として，表 6 の利用者観測データを入力した場合の OD データの生成を考える．OD データ生成の例を図 15 に示す．左のデータは，利用者観測データから観測開始時刻が 12:00:00，観測終了時刻が 12:15:00 のデータを抜粋したものである．また，右の図はあるバスの運行を示しており，この運行はバス停 1 を起点，バス停 7 を終点としている．図 15 に示すように，観測開始時刻の 12:00:00 はバス停 1 の出発時刻と合致しており，観測終了時刻の 12:15:00 はバス停 4 の出発時刻と合致しているため，バス停 1–バス停 4 の利用者としてカウントする．最終的な出力として，表 7 に示すような OD データが得られる．

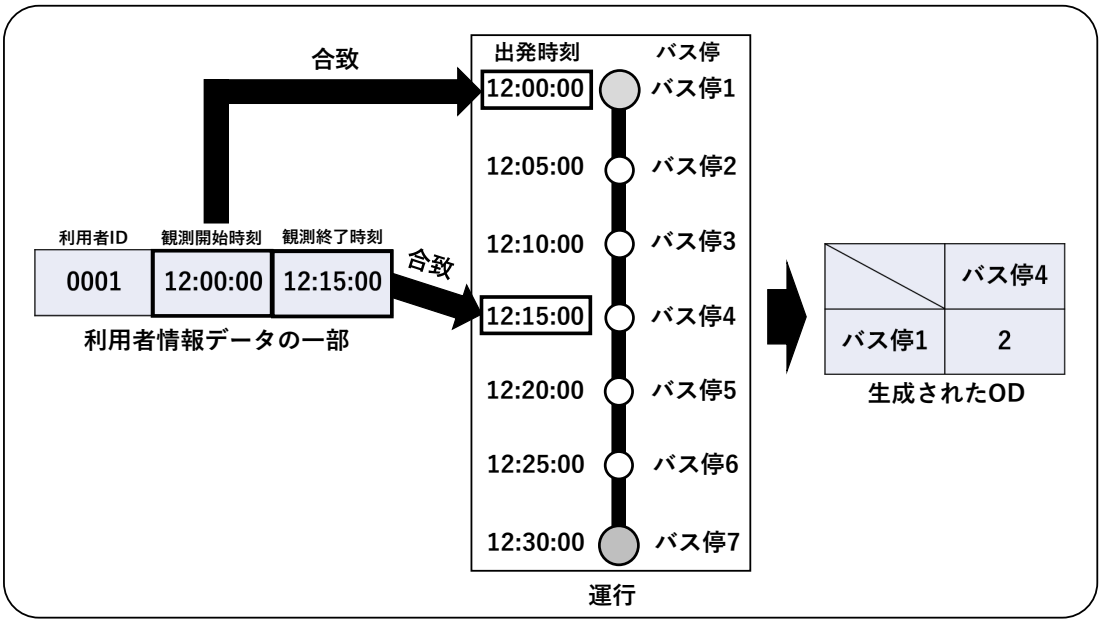

図 15: OD データ生成の例

表 7: OD データの例

降車バス停 / 乗車バス停	…	バス停 4	バス停 5	…
バス停 1	…	2	0	…
⋮	…	…	…	…
バス停 3	…	0	1	…
⋮	…	…	…	…

5. 評価実験

5.1 評価に用いるデータセットを収集する路線

本評価で用いるデータセットを収集する路線として，図 16 に示すような京都府京都市内のバス路線を選定した．本路線は，苔寺・すず虫寺〜京都駅前間を運行する約 9.3km の路線である．時刻表では，バスが始点から出発して終点へ到着するまで 55 分間かかることが想定されている．また，バス停の数が 33 であることから，最大で計 32 区間の走行時間と 31 回の各バス停での停車時間が存在する．

図 16: 評価対象路線の経路図（地図データ: Google マップ）

本路線は，下記の 2 つの要件を満たす点で，提案手法の評価に用いるデータセットを収集する対象として適している．

十分な運行時間を有すること

提案手法の妥当性を検証するためには，MAC アドレスの変更が行われる程度に利用者の乗車時間が長い必要がある．十分な運行時間を有する路線は，乗車時間が長い利用者が多くなると考えるため，提案手法の妥当性の検証での利用に適する．仮に，十分な運行時間を有していない路線の場合，利用者の乗車時間が短くなり，MAC アドレスが変更される前に降車する利用

者が多く存在することになる．この場合，アドレス・キャリーオーバーが行われず，提案手法の妥当性を示すことができない．

停留するバス停数が多いこと

停留するバス停数が多い場合，各利用者が異なるバス停区間を乗車すると考えられる．つまり，利用者の OD パターンの組み合わせが多数存在すると考えられるため，評価対象としている．

5.2 評価で用いるデータセット

5.2.1 評価で用いるデータセットの収集方法

本節では，5.1 節で記述した路線で収集したデータセットについて説明する．本評価では，以下に示す 4 運行分の BLE 広告パケットをデータセットとして使用した．8 時 37 分–9 時 33 分， 11 時 10 分–12 時 07 分の運行は，始発停留所が京都駅前，終着停留所が苔寺・すず虫寺であり，10 時 05 分–11 時 00 分，16 時 25 分–17 時 20 分の運行は，始発停留所が苔寺・すず虫寺，終着停留所が京都駅前である．

- 2022 年 12 月 23 日 8 時 37 分–9 時 33 分の BLE 広告パケット
- 2022 年 12 月 23 日 10 時 05 分–11 時 00 分の BLE 広告パケット
- 2022 年 12 月 23 日 11 時 10 分–12 時 07 分の BLE 広告パケット
- 2022 年 12 月 23 日 16 時 25 分–17 時 20 分の BLE 広告パケット

データセットとなる BLE の広告パケットは，路線バスが搭載するデジタルタコグラフ車載器に Bluetooth 通信モジュールを取り付け，hcidump を用いることで収集した．また，Bluetooth 通信モジュールの障害によってデータセットを収集できない場合を考慮し，4 章の検証実験で使用した Raspberry Pi 3 Model B+ も使用し，同日の同時間帯でデータセットの収集を行った．このようにすることで冗長性を確保している．

各データセット収集デバイスの配置を図17に示す．図17に示すように，デジタルタコグラフ車載器は運転席のほぼ真上に設置しており，Raspberry Pi 3 Model B+は最後部の左座席に設置している．

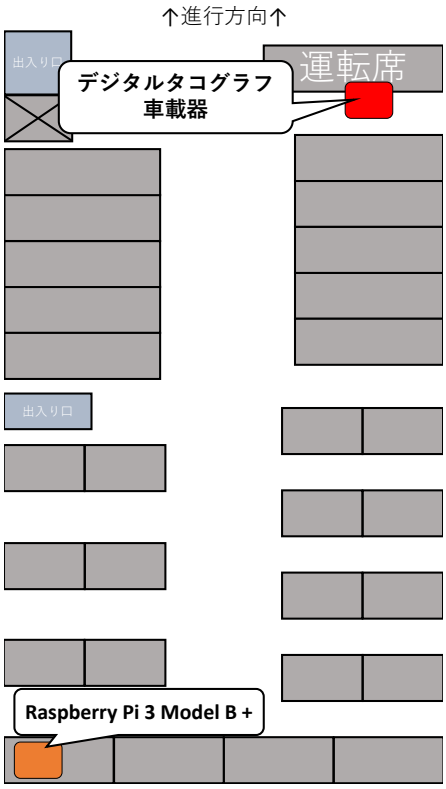

図 17: データセット収集デバイスの配置図

5.2.2 比較対象となる正解 OD データ

今回の評価では，1章で述べた目視によってODデータを記録する手法により，正解となるODデータを作成した．まず初めに，図18に示すようなバスの座席位置の図と，表8に示すような行がバス停，列が利用者である利用者の乗降バス停表を用いて記録を行った．記録方法としては，表8に示すように，利用者の乗車時，利用者と停留しているバス停の交差部分に各座席を示すアルファベットを記入し，利用者が乗車している間は，停留した各バス停でアルファベットを記入し，利用者の降車時に ◯ を記入する方法で行った．

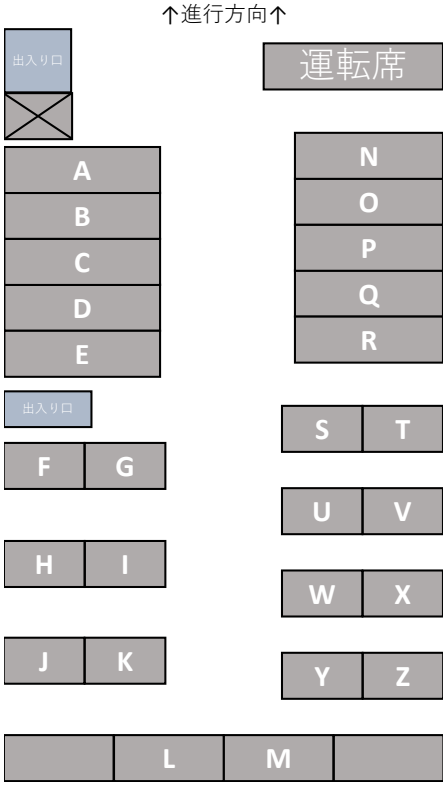

図 18: バスの座席位置の図

表 8: 利用者の乗降バス停表の例

	利用者 1	利用者 2	...	利用者 9	利用者 10
バス停 1	N		...		
バス停 2	N		...		J
バス停 3	N		...		J
バス停 4	N	U	...	V	J
バス停 5	N	U	...	V	J
⋮	⋮	⋮	⋮	⋮	⋮
バス停 16	N	U	...	V	◯
バス停 17	N	◯	...	V	
バス停 18	N		...	V	
バス停 19	N		...	◯	
バス停 20	◯		...		

この利用者の乗降バス停表を基に作成した全ての正解 OD データを図 19，図 20，図 21，図 22 に示す（利用者が存在したバス停区間を橙色，見落としにより利用者の存在が定かでないバス停区間を青色に色付けしており，利用者が乗降しないバス停区間を空欄にしている）．図 19 は 8 時 37 分–9 時 33 分の運行における正解 OD データ，図 21 は 10 時 05 分–11 時 00 分の運行における正解 OD データ，図 20 は 11 時 10 分–12 時 07 分の運行における正解データ，図 22 は 16 時 25 分–17 時 20 分の運行における正解 OD データである．正解 OD データの傾向として，乗車するバス停にほぼ偏りはみられないが，京都駅前のバス停で降車する利用者が明らかに多いことがわかる．これは，京都駅から電車やタクシーといった他の公共交通機関に乗り換えを行う利用者が多数存在するためであると考えられる．なお，この OD データは一般の利用者の OD を記録したものであるため，利用者が持つスマートフォンの Bluetooth が ON になっているか，利用者が COCOA を使用しているかは不明である．

乗車バス停＼降車バス停	烏丸七条	烏丸五条	烏丸松原	四条烏丸	四条大宮	壬生寺道	四条中新道	西大路四条	西大路三条	三条春日	山ノ内	庚申前	太秦天神川駅前	太秦東口	太秦広隆寺前	太秦開町	帷子ノ辻	嵯峨野秋街道町	生田口	有栖川［京都］	車折神社前	下嵯峨	角倉町	嵐山［京都］	嵐山公園	阪急嵐山駅前	谷ヶ辻町	内田町	松尾大社前	松室北河原町	松尾小学校前	苔寺・すず虫寺
京都駅前	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	2
烏丸七条		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
烏丸五条			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
烏丸松原				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条烏丸					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条大宮						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
壬生寺道							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条中新道								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
西大路四条									0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
西大路三条										0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
三条春日											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
山ノ内												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
庚申前													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦天神川駅前														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦東口															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦広隆寺前																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦開町																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
帷子ノ辻																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵯峨野秋街道町																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
生田口																				0	0	0	0	0	0	0	0	0	0	0	0	0
有栖川［京都］																					0	0	0	0	0	0	0	0	0	0	0	0
車折神社前																						0	0	0	0	0	0	0	0	0	0	0
下嵯峨																							0	0	0	0	0	0	0	0	0	0
角倉町																								0	0	0	0	0	0	0	0	0
嵐山［京都］																									0	0	0	0	0	0	0	0
嵐山公園																										0	0	0	0	0	0	0
阪急嵐山駅前																											0	0	0	0	0	0
谷ヶ辻町																												0	0	0	0	0
内田町																													0	0	0	1
松尾大社前																														0	0	1
松室北河原町																															0	0
松尾小学校前																																0

図 19: 8 時 37 分–9 時 33 分の運行における正解 OD データ

乗車バス停＼降車バス停	松尾小学校前	松室北河原町	松尾大社前	内田町	谷ヶ辻町	阪急嵐山駅前	嵐山公園	嵐山［京都］	角倉町	下嵯峨	車折神社前	有栖川［京都］	生田口	嵯峨野秋街道町	帷子ノ辻	太秦開町	太秦広隆寺前	太秦東口	太秦天神川駅前	庚申前	山ノ内	三条春日	西大路三条	西大路四条	四条中新道	壬生寺道	四条大宮	四条烏丸	烏丸松原	烏丸五条	烏丸七条	京都駅前
苔寺・すず虫寺	0	0	0	0	0	1	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	4
松尾小学校前		0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
松室北河原町			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
松尾大社前				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
内田町					0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
谷ヶ辻町						0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
阪急嵐山駅前							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
嵐山公園								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵐山［京都］									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
角倉町										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
下嵯峨											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
車折神社前												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
有栖川［京都］													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
生田口														0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
嵯峨野秋街道町															0	1	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0
帷子ノ辻																0	0	0	1	0	0	0	0	0	0	0	0	3	0	0	0	1
太秦開町																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦広隆寺前																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦東口																			0	0	0	0	0	0	0	0	0	0	0	0	0	1
太秦天神川駅前																				0	0	0	0	1	0	0	0	1	0	0	0	0
庚申前																					0	0	0	0	0	0	0	0	0	0	0	0
山ノ内																						0	0	0	0	0	0	0	0	0	0	0
三条春日																							0	0	0	0	0	0	0	0	0	0
西大路三条																								0	0	0	0	0	0	0	0	0
西大路四条																									0	0	0	0	0	0	0	0
四条中新道																										0	0	0	0	0	0	0
壬生寺道																											0	3	0	0	0	0
四条大宮																												0	0	0	0	0
四条烏丸																													0	0	0	0
烏丸松原																														1	0	0
烏丸五条																															0	0
烏丸七条																																0

図 20: 10 時 05 分–11 時 00 分の運行における正解 OD データ

乗車バス停＼降車バス停	烏丸七条	烏丸五条	烏丸松原	四条烏丸	四条大宮	壬生寺道	四条中新道	西大路四条	西大路三条	三条春日	山ノ内	庚申前	太秦天神川駅前	太秦東口	太秦広隆寺前	太秦開町	帷子ノ辻	嵯峨野秋街道町	生田口	有栖川［京都］	車折神社前	下嵯峨	角倉町	嵐山［京都］	嵐山公園	阪急嵐山駅前	谷ヶ辻町	内田町	松尾大社前	松室北河原町	松尾小学校前	苔寺・すず虫寺
京都駅前	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	2	0	0	0	0	0	0	0	1	0	1	2
烏丸七条		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
烏丸五条			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
烏丸松原				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条烏丸					0	0	0	0	0	0	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条大宮						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
壬生寺道							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条中新道								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
西大路四条									0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
西大路三条										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
三条春日											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
山ノ内												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
庚申前													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦天神川駅前														1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
太秦東口															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦広隆寺前																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦開町																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
帷子ノ辻																		0	0	0	0	0	0	0	0	0	1	0	2	0	0	0
嵯峨野秋街道町																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
生田口																				0	0	0	0	0	0	0	0	0	0	0	0	0
有栖川［京都］																					0	0	0	0	0	0	0	0	0	0	0	0
車折神社前																						0	0	0	0	0	0	0	0	0	0	0
下嵯峨																							0	0	0	0	0	0	0	0	0	0
角倉町																								0	0	0	0	0	0	0	0	0
嵐山［京都］																									0	0	0	0	0	0	0	0
嵐山公園																										0	0	0	0	0	0	0
阪急嵐山駅前																											0	0	0	0	0	0
谷ヶ辻町																												0	0	0	0	0
内田町																													0	0	0	0
松尾大社前																														0	0	1
松室北河原町																															0	0
松尾小学校前																																0

図 21: 11 時 10 分–12 時 07 分の運行における正解 OD データ

乗車バス停＼降車バス停	松尾小学校前	松室北河原町	松尾大社前	内田町	谷ヶ辻町	阪急嵐山駅前	嵐山公園	嵐山［京都］	角倉町	下嵯峨	車折神社前	有栖川［京都］	生田口	嵯峨野秋街道町	帷子ノ辻	太秦開町	太秦広隆寺前	太秦東口	太秦天神川駅前	庚申前	山ノ内	三条春日	西大路三条	西大路四条	四条中新道	壬生寺道	四条大宮	四条烏丸	烏丸松原	烏丸五条	烏丸七条	京都駅前
苔寺・すず虫寺	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
松尾小学校前		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
松室北河原町			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
松尾大社前				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
内田町					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
谷ヶ辻町						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
阪急嵐山駅前							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵐山公園								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
嵐山［京都］									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
角倉町										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
下嵯峨											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
車折神社前												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
有栖川［京都］													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
生田口														0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	1
嵯峨野秋街道町															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
帷子ノ辻																0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
太秦開町																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦広隆寺前																		0	0	0	0	0	0	0	0	0	0	0	1	0	0
太秦東口																			0	0	0	0	0	0	0	0	0	0	0	0	0
太秦天神川駅前																				0	0	0	0	0	0	0	0	0	0	0	0
庚申前																					0	0	0	0	0	0	0	1	0	0	0
山ノ内																						0	0	0	0	0	0	0	0	0	0
三条春日																							0	0	0	0	0	0	0	0	0
西大路三条																								0	0	0	0	0	0	0	0
西大路四条																									0	0	1	0	0	0	0
四条中新道																										0	0	0	0	0	0
壬生寺道																											0	0	0	0	0
四条大宮																												0	0	0	0
四条烏丸																													0	0	3
烏丸松原																														0	1
烏丸五条																															1
烏丸七条																															0

図 22: 16 時 25 分–17 時 20 分における正解 OD データ

5.3 評価方法

本評価では，提案システムが生成する OD データの精度を検証する．そのために，事前に収集したデータセットを入力とし，4.2 節で述べた提案システムを用いて OD データを取得する実験を行う．上述する実験で取得した推定 OD データと正解 OD データを比較することで，OD データの精度を示す．本評価は，各運行において，停車せずに通過したバス停（通過バス停）を除外しない場合と除外した場合について評価した．通過バス停で乗車もしくは降車した利用者の OD を除外することで，OD データの取得精度が向上すると考えたためである．

精度の比較では，以下に示す適合率，再現率を用いた．ここで，OD は各利用者の乗車バス停と降車バス停の組を示す．適合率は，推定 OD データにおける OD の総数（推定 OD 数）に対する推定 OD データにおける OD の正解数（推定 OD の正解数）の割合である（式 (1) 参照）．また，再現率は正解 OD データにおける OD の総数（正解 OD 数）に対する推定 OD の正解数の割合である（式 (2) 参照）．通常，OD データの精度評価は再現率のみを用いることが多いが，提案手法は COCOA の広告パケットを用いるため，COCOA の使用が不明なバスの全利用者の OD 数を正解 OD 数とする再現率のみの評価は適切でないと考えられる．したがって，COCOA の広告パケットから推定した OD 数を推定 OD 数とする適合率を評価指標に加えている．

適合率と再現率の算出に関して，図 23 に示すように，各利用者の推定 OD と正解 OD が完全一致した場合にのみ正解とする．また，図 24 に示すように，推定 OD の乗車バス停（もしくは降車バス停）が正解 OD と一致しない場合（不正解 1）や，推定 OD と合致する正解 OD が存在しない場合（不正解 2），正解 OD と合致する推定 OD が存在しない場合は不正解とする．

$$\text{適合率} = \frac{\text{推定 OD の正解数}}{\text{推定 OD 数}} \tag{1}$$

$$\text{再現率} = \frac{\text{推定 OD の正解数}}{\text{正解 OD 数}} \tag{2}$$

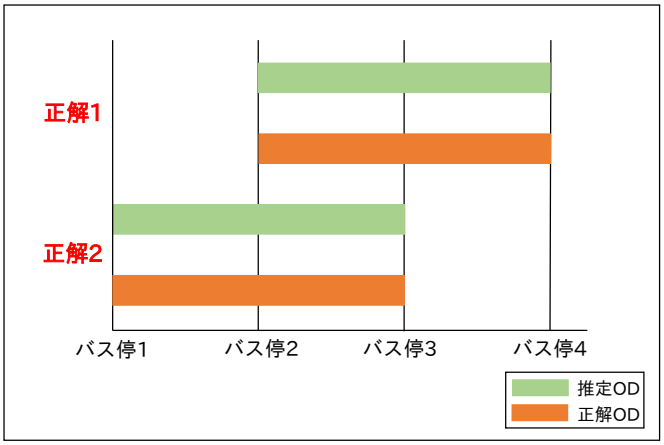

図 23: 評価における正解例

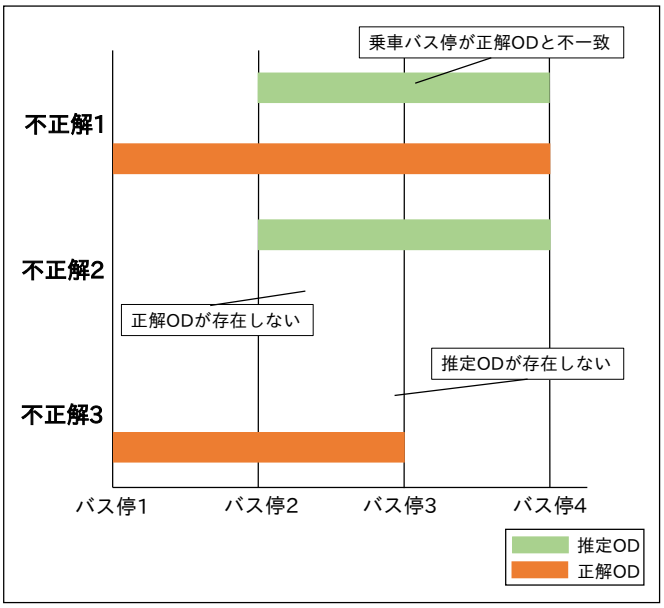

図 24: 評価における不正解例

5.4 パラメータの設定

この章では，4.1 節で述べた路線バス用のアドレス・キャリーオーバー・アルゴリズムを用いる上で必要となる各パラメータの設定について述べる．

設定が必要となるパラメータは$rssi_th$, $passenget\_th$, $extract\_int$, $timetable_int$の 4 つである．次に，各パラメータの設定について述べる．

$rssi\_th$ に関して，バス利用者のスマートフォンが発する MAC アドレスに対する RSSI の分布を図 25 に示す．図 25 から，RSSI が −65 dB〜−80 dB の範囲に収束しているため，$rssi\_th$ を 15 dB に設定した．

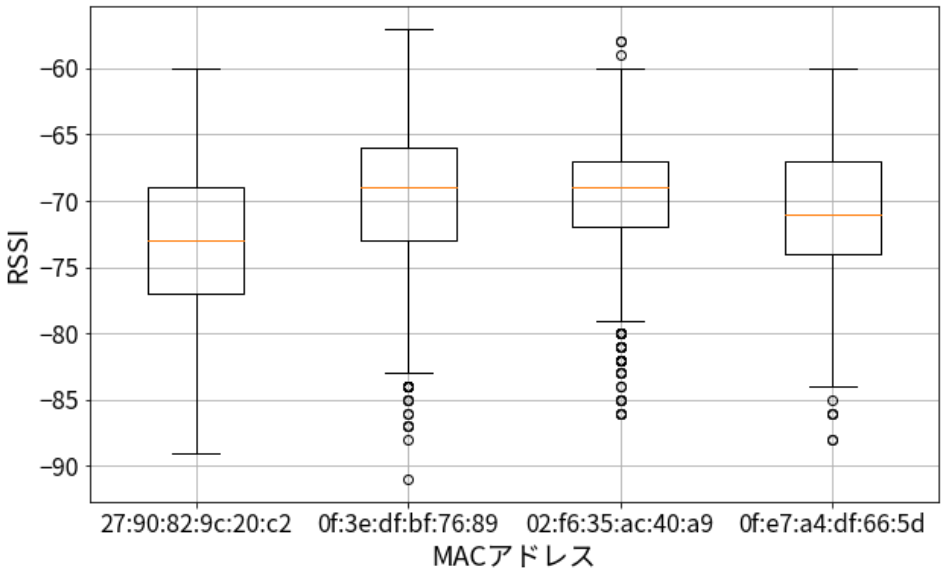

図 25: MAC アドレスに対する RSSI の分布

$passenger\_th$ に関して，評価対象路線では，バス停区間の最短所用時間（バス停区間の走行時間とバス停における停車時間を足し合わせた時間）が 60 秒である．この 60 秒を超えず，バスの早着により利用者が早く降車した場合を考慮して，$passenger\_th$ を 50 秒に設定した．つまり，50 秒以上観測された MAC アドレスをバスの利用者とした．

$extract\_int$ に関して，以下の 4 運行分の BLE 広告パケットをデータセットとし

て使用し，*extract_int* を 10 秒から 60 秒まで変化させたときの適合率を調査した．

- 2022 年 12 月 23 日 8 時 37 分–9 時 33 分の BLE 広告パケット
- 2022 年 12 月 23 日 10 時 05 分–11 時 00 分の BLE 広告パケット
- 2022 年 12 月 23 日 11 時 10 分–12 時 07 分の BLE 広告パケット
- 2022 年 12 月 23 日 16 時 25 分–17 時 20 分の BLE 広告パケット

調査結果を図 26 に示す．図 26 の青，緑，紫，黒のグラフは 8 時 37 分–9 時 33 分の BLE 広告パケット，10 時 05 分–11 時 00 分の BLE 広告パケット，11 時 10 分–12 時 07 分の BLE 広告パケット，16 時 25 分–17 時 20 分の BLE 広告パケットを入力とした時の適合率の変化を示しており，赤のグラフは上記の 4 運行の適合率の平均を示している．図 26 から，*extract_int* が 50〜60 秒のときに適合率の平均が最も高くなったため，*extract_int* を 50 秒に設定した．

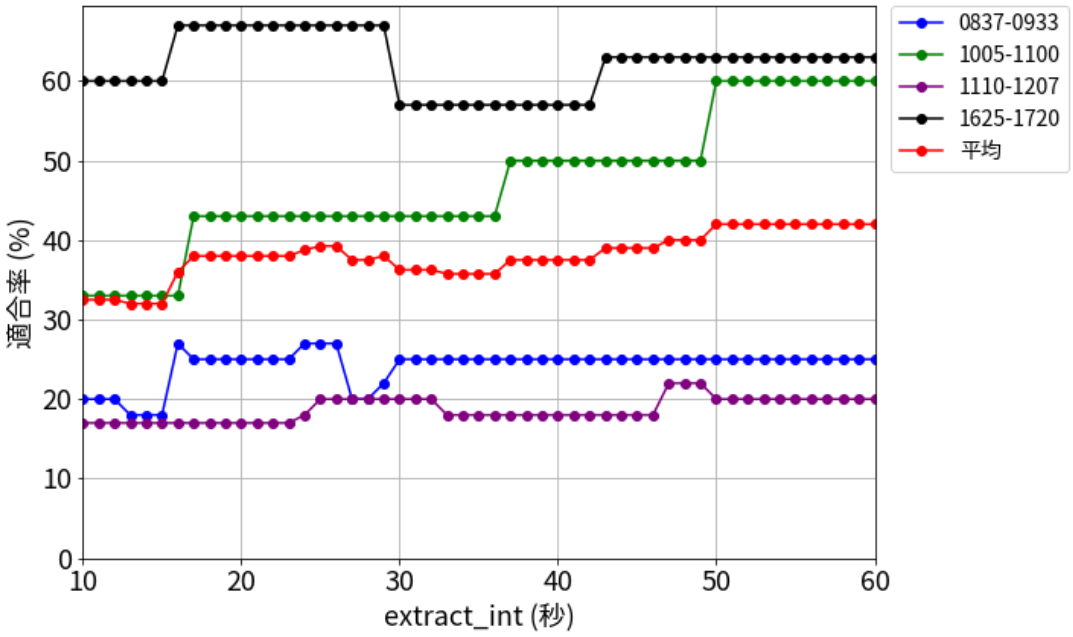

図 26: *extract_int* の設定

timetable_int に関して，評価対象路線では，バス停区間の最短所用時間が 60 秒である．この 60 秒を超えず，バスの早着や遅発を考慮して *timetable_int* を 20

秒に設定した．つまり，時刻表の発車時刻の ±20 秒以内に観測された MAC アドレスをバスの利用者とした．

5.5 評価結果

5.5.1 8 時 37 分–9 時 33 分の運行における推定 OD データ

8 時 37 分–9 時 33 分の運行における推定 OD データを図 27 に示す．この推定 OD データは，デジタルタコグラフ車載器により収集した BLE 広告パケットをデータセットとして使用した．本提案システムにより，利用者が存在したと推定されたバス停区間は，京都駅前–烏丸七条，京都駅前–烏丸五条，京都駅前–苔寺・すず虫寺，烏丸七条–烏丸五条，西大路三条–山の内，有栖川 [京都]–車折神社前の 6 区間となった．また，各区間の人数は 1 名，1 名，2 名，2 名，1 名，1 名であり，利用者の総数は 8 名と推定された．その中で，正解 OD データと比較した際，正しく推定されていた区間は京都駅前–苔寺・すず虫寺であった．また，正しく推定されていた区間の人数は 2 名であるため，提案システムで取得された OD データの正解数は 2 名となった．

5.5.2 10 時 05 分–11 時 00 分の運行における推定 OD データ

10 時 05 分–11 時 00 分の運行における推定 OD データを図 28 に示す．この推定 OD データは，デジタルタコグラフ車載器により収集した BLE 広告パケットをデータセットとして使用した．本提案システムにより，利用者が存在したと推定されたバス停区間は，苔寺・すず虫寺–京都駅前，阪急嵐山駅–嵐山 [京都]，帷子ノ辻–京都駅前，四条烏丸–烏丸松原の 4 区間となった．また，各区間の人数は 2 名，1 名，1 名，1 名であり，利用者の総数は 5 名と推定された．その中で，正解 OD データと比較した際，正しく推定されていた区間は苔寺・すず虫寺–京都駅前，帷子ノ辻–京都駅前であった．また，正しく推定されていた各区間の人数は 2 名，1 名であり，提案システムで取得された OD データの正解数は 3 名となった．

降車バス停 / 乗車バス停	烏丸七条	烏丸五条	烏丸松原	四条烏丸	四条大宮	壬生寺道	四条中新道	西大路四条	西大路三条	三条春日	山ノ内	庚申前	太秦天神川駅前	太秦東口	太秦広隆寺前	太秦開町	帷子ノ辻	嵯峨野秋街道町	生田口	有栖川［京都］	車折神社前	下嵯峨	角倉町	嵐山［京都］	嵐山公園	阪急嵐山駅前	谷ヶ辻町	内田町	松尾大社前	松室北河原町	松尾小学校前	苔寺・すず虫寺
京都駅前	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
烏丸七条		2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
烏丸五条			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
烏丸松原				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条烏丸					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条大宮						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
壬生寺道							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条中新道								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
西大路四条									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
西大路三条										0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
三条春日											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
山ノ内												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
庚申前													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦天神川駅前														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦東口															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦広隆寺前																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦開町																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
帷子ノ辻																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵯峨野秋街道町																			0	0	0	0	0	0	0	0	0	0	0	0	0
生田口																				0	0	0	0	0	0	0	0	0	0	0	0
有栖川［京都］																					1	0	0	0	0	0	0	0	0	0	0
車折神社前																						0	0	0	0	0	0	0	0	0	0
下嵯峨																							0	0	0	0	0	0	0	0	0
角倉町																								0	0	0	0	0	0	0	0
嵐山［京都］																									0	0	0	0	0	0	0
嵐山公園																										0	0	0	0	0	0
阪急嵐山駅前																											0	0	0	0	0
谷ヶ辻町																												0	0	0	0
内田町																													0	0	0
松尾大社前																														0	0
松室北河原町																															0
松尾小学校前																															0

図 27: 8 時 37 分–9 時 33 分の推定 OD データ

降車バス停 / 乗車バス停	松尾小学校前	松室北河原町	松尾大社前	内田町	谷ヶ辻町	阪急嵐山駅前	嵐山公園	嵐山［京都］	角倉町	下嵯峨	車折神社前	有栖川［京都］	生田口	嵯峨野秋街道町	帷子ノ辻	太秦開町	太秦広隆寺前	太秦東口	太秦天神川駅前	庚申前	山ノ内	三条春日	西大路三条	西大路四条	四条中新道	壬生寺道	四条大宮	四条烏丸	烏丸松原	烏丸五条	烏丸七条	京都駅前
苔寺・すず虫寺	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
松尾小学校前		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
松室北河原町			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
松尾大社前				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
内田町					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
谷ヶ辻町						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
阪急嵐山駅前							0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵐山公園								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵐山［京都］									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
角倉町										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
下嵯峨											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
車折神社前												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
有栖川［京都］													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
生田口														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵯峨野秋街道町															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
帷子ノ辻																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
太秦開町																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦広隆寺前																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦東口																			0	0	0	0	0	0	0	0	0	0	0	0	0
太秦天神川駅前																				0	0	0	0	0	0	0	0	0	0	0	0
庚申前																					0	0	0	0	0	0	0	0	0	0	0
山ノ内																						0	0	0	0	0	0	0	0	0	0
三条春日																							0	0	0	0	0	0	0	0	0
西大路三条																								0	0	0	0	0	0	0	0
西大路四条																									0	0	0	0	0	0	0
四条中新道																										0	0	0	0	0	0
壬生寺道																											0	0	0	0	0
四条大宮																												0	0	0	0
四条烏丸																													1	0	0
烏丸松原																														0	0
烏丸五条																															0
烏丸七条																															

図 28: 10 時 05 分–11 時 00 分の推定 OD データ

5.5.3 11 時 10 分–12 時 07 分の運行における推定 OD データ

11 時 10 分–12 時 07 分の運行における推定 OD データを図 29 に示す．この推定 OD データは，デジタルタコグラフ車載器により収集した BLE 広告パケットをデータセットとして使用した．本提案システムにより，利用者が存在したと推定されたバス停区間は，京都駅前–壬生寺道，京都駅前–苔寺・すず虫寺，西大路四条–西大路三条，太秦広隆寺前–帷子ノ辻，太秦開町–帷子ノ辻，有栖川 [京都]–下嵯峨，有栖川 [京都]–角倉町の 7 区間となった．また，各区間の人数は 1 名，2 名，1 名，1 名，3 名，1 名，1 名であり，利用者の総数は 10 名と推定された．その中で，正解 OD データと比較した際，正しく推定されていた区間は苔寺・すず虫寺–京都駅前であった．また，正しく推定されていた区間の人数は 2 名であるため，提案システムで取得された OD データの正解数は 2 名となった．

5.5.4 16 時 25 分–17 時 20 分の運行における推定 OD データ

11 時 10 分–12 時 07 分の運行における推定 OD データを図 30，図 31 に示す．図 30 の推定 OD データは，デジタルタコグラフ車載器により収集した BLE 広告パケットをデータセットとして使用した．また，図 31 の推定 OD データは，Raspberry Pi 3 Model B+により収集した BLE 広告パケットをデータセットとして使用した．

図 30 の推定 OD データに関して，本提案システムにより，利用者が存在したと推定されたバス停区間は，苔寺・すず虫寺–京都駅前，嵐山 [京都]–西大路三条，生田口–京都駅前，帷子ノ辻–京都駅前，西大路三条–西大路四条，四条中新道–壬生寺道の 6 区間となった．また，各区間の人数は，3 名，1 名，1 名，1 名，1 名，1 名であり，利用者の総数は 8 名と推定された．その中で，正解 OD データと比較した際，正しく推定されていた区間は苔寺・すず虫寺–京都駅前，生田口–京都駅前，帷子ノ辻–京都駅前であった．また，正しく推定されていた各区間の人数は 3 名，1 名，1 名であり，提案システムで取得された OD データの正解数は 5 名となった．

図 31 の推定 OD データに関して，本提案システムにより，利用者が存在したと推定されたバス停区間は，京都駅前–苔寺・すず虫寺，有栖川 [京都]–京都駅前，

乗車バス停＼降車バス停	烏丸七条	烏丸五条	烏丸松原	四条烏丸	四条大宮	壬生寺道	四条中新道	西大路四条	西大路三条	三条春日	山ノ内	庚申前	太秦天神川駅前	太秦東口	太秦広隆寺前	太秦開町	帷子ノ辻	嵯峨野秋街道町	生田口	有栖川［京都］	車折神社前	下嵯峨	角倉町	嵐山［京都］	嵐山公園	阪急嵐山駅前	谷ヶ辻町	内田町	松尾大社前	松室北河原町	松尾小学校前	苔寺・すず虫寺
京都駅前	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
烏丸七条		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
烏丸五条			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
烏丸松原				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条烏丸					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条大宮						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
壬生寺道							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
四条中新道								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
西大路四条									1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
西大路三条										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
三条春日											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
山ノ内												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
庚申前													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦天神川駅前														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦東口															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦広隆寺前																0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦開町																	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
帷子ノ辻																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵯峨野秋街道町																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
生田口																				0	0	0	0	0	0	0	0	0	0	0	0	0
有栖川［京都］																					0	1	1	0	0	0	0	0	0	0	0	0
車折神社前																						0	0	0	0	0	0	0	0	0	0	0
下嵯峨																							0	0	0	0	0	0	0	0	0	0
角倉町																								0	0	0	0	0	0	0	0	0
嵐山［京都］																									0	0	0	0	0	0	0	0
嵐山公園																										0	0	0	0	0	0	0
阪急嵐山駅前																											0	0	0	0	0	0
谷ヶ辻町																												0	0	0	0	0
内田町																													0	0	0	0
松尾大社前																														0	0	0
松室北河原町																															0	0
松尾小学校前																																0

図 29: 11 時 10 分–12 時 07 分の推定 OD データ

生田口–烏丸松原，西大路四条–西大路三条，四条中新道–壬生寺道の 5 区間となった．また，各区間の人数は 3 名，1 名，1 名，2 名，2 名であり，利用者の総数は 9 名と推定された．その中で，正解 OD データと比較した際，正しく推定されていた区間は苔寺・すず虫寺–京都駅前であった．また，正しく推定されていた各区間の人数は 3 名であるため，提案システムで取得された OD データの正解数は 3 名となった．

降車バス停 / 乗車バス停	松尾小学校前	松室北河原町	松尾大社前	内田町	谷ヶ辻町	阪急嵐山駅前	嵐山公園	嵐山［京都］	角倉町	下嵯峨	車折神社前	有栖川［京都］	生田口	嵯峨野秋街道町	帷子ノ辻	太秦開町	太秦広隆寺前	太秦東口	太秦天神川駅前	庚申前	山ノ内	三条春日	西大路三条	西大路四条	四条中新道	壬生寺道	四条大宮	四条烏丸	烏丸松原	烏丸五条	烏丸七条	京都駅前
苔寺・すず虫寺	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
松尾小学校前		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
松室北河原町			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
松尾大社前				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
内田町					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
谷ヶ辻町						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
阪急嵐山駅前							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵐山公園								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵐山［京都］									0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
角倉町										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
下嵯峨											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
車折神社前												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
有栖川［京都］													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
生田口														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
嵯峨野秋街道町															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
帷子ノ辻																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
太秦開町																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦広隆寺前																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦東口																			0	0	0	0	0	0	0	0	0	0	0	0	0
太秦天神川駅前																				0	0	0	0	0	0	0	0	0	0	0	0
庚申前																					0	0	0	0	0	0	0	0	0	0	0
山ノ内																						0	0	0	0	0	0	0	0	0	0
三条春日																							0	0	0	0	0	0	0	0	0
西大路三条																								1	0	0	0	0	0	0	0
西大路四条																									0	0	0	0	0	0	0
四条中新道																										1	0	0	0	0	0
壬生寺道																											0	0	0	0	0
四条大宮																												0	0	0	0
四条烏丸																													0	0	0
烏丸松原																														0	0
烏丸五条																															0
烏丸七条																															0

図 30: 16 時 25 分–17 時 20 分の推定 OD データ（デジタルタコグラフ車載器）

5.5.5 通過バス停を除外しない場合の OD データ取得精度

各運行において通過バス停を除外しない場合の OD データ取得精度を表 9 に示す．運行時間が 16:25–17:20 のとき，適合率と再現率が最も高い結果となった．各

乗車バス停＼降車バス停	松尾小学校前	松室北河原町	松尾大社前	内田町	谷ヶ辻町	阪急嵐山駅前	嵐山公園	嵐山［京都］	角倉町	下嵯峨	車折神社前	有栖川［京都］	生田口	嵯峨野秋街道町	帷子ノ辻	太秦開町	太秦広隆寺前	太秦東口	太秦天神川駅前	庚申前	山ノ内	三条春日	西大路三条	西大路四条	四条中新道	壬生寺道	四条大宮	四条烏丸	烏丸松原	烏丸五条	烏丸七条	京都駅前
苔寺・すず虫寺	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
松尾小学校前		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
松室北河原町			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
松尾大社前				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
内田町					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
谷ヶ辻町						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
阪急嵐山駅前							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵐山公園								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
嵐山［京都］									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
角倉町										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
下嵯峨											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
車折神社前												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
有栖川［京都］													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
生田口														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
嵯峨野秋街道町															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
帷子ノ辻																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦開町																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦広隆寺前																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
太秦東口																			0	0	0	0	0	0	0	0	0	0	0	0	0
太秦天神川駅前																				0	0	0	0	0	0	0	0	0	0	0	0
庚申前																					0	0	0	0	0	0	0	0	0	0	0
山ノ内																						0	0	0	0	0	0	0	0	0	0
三条春日																							0	0	0	0	0	0	0	0	0
西大路三条																								2	0	0	0	0	0	0	0
西大路四条																									0	0	0	0	0	0	0
四条中新道																										2	0	0	0	0	0
壬生寺道																											0	0	0	0	0
四条大宮																												0	0	0	0
四条烏丸																													0	0	0
烏丸松原																														0	0
烏丸五条																															0
烏丸七条																															0

図 31: 16 時 25 分–17 時 20 分の推定 OD データ（Raspberry Pi 3 Model B+）

収集デバイスを比較した際，適合率，再現率ともに，デジタルタコグラフ車載器を収集デバイスとして用いた場合の方が高い結果となった．

表 9: 通過バス停を除外しない場合の OD データ取得精度

	運行時間	適合率	再現率
デジタルタコグラフ車載器	8:37–9:33	25%	17%
	10:05–11:00	60%	-
	11:10–12:07	20%	10%
	16:25–17:20	63%	22%
Raspberry Pi 3 Model B+	16:25–17:20	33%	13%

5.5.6 通過バス停を除外した場合の OD データ取得精度

各運行において通過バス停を除外した場合の OD データ取得精度を表 10 に示す．運行時間が 8:37–9:33，16:25–17:20 のとき，適合率と再現率が最も高い結果となった．各収集デバイスを比較した際，適合率，再現率ともに，デジタルタコグラフ車載器を収集デバイスとして用いた場合の方が高い結果となった．表 9 と表 10 の比較により，適合率は最大で 75%，再現率は最大で 5%向上した．

表 10: 通過バス停を除外した場合の OD データ取得精度

	運行時間	適合率	再現率
デジタルタコグラフ車載器	8:37–9:33	100%	22%
	10:05–11:00	60%	-
	11:10–12:07	67%	10%
	16:25–17:20	100%	22%
Raspberry Pi 3 Model B+	16:25–17:20	60%	13%

6. 考察

5 章の評価実験では，京都市の路線バス京都府京都市の路線バスの実データを用いて，OD データ取得精度の比較評価を行った．本章では，評価実験で得られた結果から考察を述べ，その後，今後の展望について述べる．

6.1 OD データ取得精度の考察

適合率と再現率を用いて，OD データの取得精度を考察する．

適合率に関して，5.5.6 項の評価結果から，通過バス停を除外しない場合の適合率（通過バス停非除外の適合率）は最も高い場合に 63%，最も低い場合に 20%となった．5.5 項で述べたように，正しく推定された区間における最短区間は帷子ノ辻–京都駅前であり，この区間の所要時間は 35 分間である．3.2.3 項より，COCOA はランダム MAC アドレスを使用し，MAC アドレスの変更間隔は最長で約 20 分と推測されるため，少なくとも 1 度はアドレスが変更されていると考えられる．したがって，既存手法では，ランダム MAC アドレスを用いるスマートフォンを追跡できなかったのに対し，本提案手法は追跡できることを示した．

しかし，通過バス停非除外の適合率は，実運用に耐える精度ではないと考えられる．通過バス停非除外の適合率が低い原因として，路線バス周辺に存在する非利用者の MAC アドレスを完全に除去できなかったことが考えられる．実例として，デジタルタコグラフ車載器を用いた場合の 16:25–17:20 の運行では，誤って推定されたバス停区間は西大路三条–西大路四条，四条中新道–壬生寺道であり，各所要時間は 2 分，1 分となっている．これらの区間において，西大路四条から京都駅前にかけて道路が渋滞していたため，各区間において路線バス前方に存在する自動車の搭乗者の MAC アドレスを観測したと考えられる．つまり，自動車の搭乗者である非利用者の MAC アドレスを除去できなかったことが通過バス停非除外の適合率が低い原因と考えられる．この点に関して，5.5.6 項の評価結果から，推定した OD データから利用者の乗降のないバス停で乗車もしくは降車した利用者の OD を除外することで抑制できると考えられる．本提案では，データセットを収集する路線における全てのバス停を対象としているため，利用者の乗降のな

いバス停で乗車もしくは降車した利用者の OD が含まれているためである．

再現率が低い点に関して，バスの利用者に対する COCOA の利用者が少ないことが考えられる．総務省の資料報告（接触確認アプリ (COCOA) の活用促進について）[16] によると，2021 年 12 月 9 日時点でのダウンロード数が 2137 万件となっている．つまり，日本の総人口を 1 億 2000 万人と仮定した場合に COCOA の利用者は総人口の 17%となる．日本の総人口に対する COCOA の利用者数と再現率が概ね合致することから，バスの利用者に対する COCOA の利用者が少ないことが，再現率低下の原因であると言える．

6.2 収集位置による精度の違いについて

5.5.5 項の評価結果から，適合率と再現率の両方において，デジタルタコグラフ車載器を用いた場合の OD データより，Raspberry Pi 3 Model B+を用いた場合の OD データの方が性能が低いことがわかる．原因として，Raspberry Pi 3 Model B+を用いて BLE の広告パケットを収集する際，Raspberry Pi 3 Model B+を運転席から見て最後部の右側の座席に設置していたため，路線バス車外に存在する非利用者のスマートフォンが送信する広告パケットが，収集した広告パケットに多数含まれていたと考えられる．

6.3 今後の展望

提案手法の評価により，既存手法では，ランダム MAC アドレスを用いるスマートフォンを追跡できなかったのに対し，提案手法は追跡できることを示した．しかし，提案手法を組み込んだ OD データ自動取得システムの OD データ取得精度は低い結果となった．OD データの取得精度が低い要因は，6.1 節で述べたように，路線バス周辺に存在する非利用者の MAC アドレスを完全に除去できなかったことにある．そこで，今後の課題として，路線バスの GPS 情報や加速度等から路線バスの停車バス停を推定し，推定した停車バス停以外のバス停で乗車もしくは降車した利用者の OD を自動的に除外する手法を取り入れることで，適合率の改善を図る．

7. おわりに

路線バスの運行効率を高めるには，バス利用者の利用動向を示すODデータの活用が必要不可欠であり，人手に頼らず取得する手法が望まれる．そのため，本研究ではODデータの自動取得を目的とし，バスの利用者数，金銭コスト，利用者の携行品の観点から，スマートフォンが発するパケットを用いたODデータの自動取得手法について着目した．しかし，プライバシー保護の観点から，ランダムMACアドレスが使用されるようになって以降，ランダムMACアドレスを利用するスマートフォンに対するODデータの自動取得手法は確立されていない．

本研究では，アドレス・キャリーオーバ・アルゴリズムをベースとした，複数のスマートフォンによるMACアドレスランダム化の非同期性を用いたODデータ自動取得手法を提案した．この手法は，RSSIを識別子として用いたアドレス・キャリーオーバー・アルゴリズムにより，MACアドレスと識別子が同期的に変わる問題を解決した．また，RSSIを識別子として用いるだけではスマートフォンの追跡精度が低下するため，アドレス・キャリーオーバーの対象となるMACアドレスを制限することでスマートフォンの追跡精度を向上させた．さらに，バスの利用者と非利用者の判別や利用者の乗降の判別を行う工夫も取り入れた．

評価では，京都府京都市の路線を対象に評価実験を行った結果，通過バス停を除外しない場合，ODデータの取得精度は最も高い場合に63%，最も低い場合に20%となった．この結果は，既存手法がランダムMACアドレスを用いるスマートフォンを追跡できなかったのに対して本提案は追跡できる一方，本提案のODデータ取得精度が低いことを示した．通過バス停を除外した場合，ODデータの取得精度は最も高い場合に100%，最も低い場合に60%となったことから，今後の課題として，路線バスのGPS情報や加速度等から路線バスの停車バス停を推定し，推定した停車バス停以外のバス停で乗車もしくは降車した利用者のODを自動的に除外する手法を取り入れることで，ODデータの取得精度が向上する可能性がある．

謝辞

主指導教員であり，既存手法に対する研究の新規性や提案手法の実用性，研究の進め方など多角的視点から，適切に研究指導をしていただきました本学情報基盤システム学研究室の藤川和利教授に心から感謝いたします．副指導教員であり，路線バスが有する特徴や提案を実装する上での問題について的確な助言をいただきました本学ユビキタスコンピューティングシステム研究室の安本慶一教授に感謝の意を表します．副指導教員であり，本学情報基盤システム学研究室の新井イスマイル准教授には，研究の独自性を見出す上で必要となる関連研究の調査方法や，無線通信に関する専門的知識や技術を基に研究の方向性についてご指導いただきました．実証実験では，倫理審査等の手続きをしていただき，提案手法の評価に用いる正解データの作成にご協力くださったことも深く感謝いたします．提案手法や予備実験に関して多くのアドバイスをくださり，学内システムの利用や研究室の設備や機器などについてもご教示くださいました本学情報基盤システム学研究室の垣内正年助教に心より感謝いたします．論文執筆にあたり，遠藤新助教には，論文全体の章構成や文章表現に関する多くのご助言をいただきました．ここに感謝いたします．そして，様々な面から研究活動を支援してくださいました本学総合情報基盤センターの辻元理恵女史，中野彩子女史に心から感謝いたします．様々な事務手続きだけでなく，研究室での生活の中でたくさんのコミュニケーションをとっていただきました．実証実験を行うにあたり，みなと観光バス株式会社の皆様，京都バス株式会社の皆様には，提案手法の評価に必要なデータや実証実験環境を提供していただきました．ご協力いただきありがとうございました．そして，2 年間の博士前期課程を共に過ごした本学情報基盤システム学研究室同期の桂さん，奥村君，小松君，ならびに，本学情報基盤システム学研究室の学生の皆様には，研究に関する様々な相談に応じていただきました．心より感謝申し上げます．最後に，修士前期課程への進学にあたり，私の意思を尊重し，経済面や生活面での援助や励ましとともに 2 年間見守ってくださいました家族には，感謝の念にたえません．本当にありがとうございました．

参考文献

[1] 国土交通省, “地域交通をめぐる現状と課題,” https://www.mlit.go.jp/policy/shingikai/content/001311082.pdf, 2019.

[2] 滝沢市　都市整備部　交通政策課, “路線バス乗り込み OD 調査　調査 結 果,” https://www.city.takizawa.iwate.jp/var/rev0/0102/7936/10_hosokusiryou2-ODtyousa.pdf, 2016.

[3] 国土交通省 中部運輸局, “バスデータ活用大百科,” https://wwwtb.mlit.go.jp/kanto/content/000166077.pdf, 2020.

[4] W. Wang, J. P. Attanucci, and N. H. Wilson, “Bus Passenger Origin-Destination Estimation and Related Analyses Using Automated Data Collection Systems,” *Journal of Public Transportation*, vol. 14, pp. 131–150, 12 2011.

[5] D. Huang, J. Yu, S. Shen, Z. Li, L. Zhao, and C. Gong, “A Method for Bus OD Matrix Estimation Using Multisource Data,” *Journal of Advanced Transportation*, vol. 2020, pp. 1–13, 3 2020.

[6] R. Takao, N. Ikeuchi, H. Suzuki, and Y. Matsumoto, “A Proposal for OD Data Estimation System of Bus Users with Intelligent Video Analysis and Its Application to Synerex,” in *2021 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2021, pp. 1–4.

[7] V. Kostakos, T. Camacho, and C. Mantero, “Towards Proximity-based Passenger Sensing on Public Transport Buses,” *Personal and Ubiquitous Computing*, vol. 17, pp. 1807–1816, 7 2013.

[8] M. Dunlap, Z. Li, K. Henrickson, and Y. Wang, “Estimation of Origin and Destination Information from Bluetooth and Wi-Fi Sensing for Transit,” *Transportation Research Record*, vol. 2595, pp. 11–17, 1 2016.

[9] D. B. Paradeda, W. K. Junior, and R. C. Carlson, “Bus Passenger Counts Using Wi-Fi Signals: Some Cautionary Findings,” *TRANSPORTES*, vol. 27, pp. 115–130, 11 2019.

[10] Z. Pu, M. Zhu, Z. Cui, and Y. Wang, “Mining Public Transit Ridership Flow and Origin-Destination Information from Wi-Fi and Bluetooth Sensing Data,” *IEEE Internet of Things Journal*, vol. 8, pp. 474–486, 1 2021.

[11] J. Barcelö, L. Montero, L. Marqués, and C. Carmona, “Travel Time Forecasting and Dynamic Origin-Destination Estimation for Freeways Based on Bluetooth Traffic Monitoring,” *Transportation research record*, vol. 2175, pp. 19–27, 1 2010.

[12] Apple, “Wi-Fi のプライバシー,” https://support.apple.com/ja-jp/guide/security/secb9cb3140c/web, 2022.

[13] Android オープンソースプロジェクト, “MAC アドレスランダム化の実装,” https://source.android.com/devices/tech/connect/wifi-mac-randomization?hl=ja, 2021.

[14] J. K. Becker, D. Li, and D. Starobinski, “Tracking Anonymized Bluetooth Devices,” *Proceedings on Privacy Enhancing Technologies*, vol. 2019, pp. 50–65, 7 2019.

[15] Apple and Google, “Exposure Notification - Bluetooth Specification v1.2,” https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-BluetoothSpecificationv1.2.pdf?1, 2020.

[16] 厚生労働省, “接触確認アプリ (COCOA) の活用促進について,” https://www.mhlw.go.jp/content/10900000/000704457.pdf, 2020.