実行中のスマートフォンアプリが扱う情報をリアルタイムに可視化する開発環境の構築

坂本 大輔[1] 原崎 將吾[2] 村尾 和哉[2] 望月 祐洋[3] 西尾 信彦[2]

概要：スマートフォンのさまざまなセンサを用いてユーザの測位や行動認識を行い，状況に応じたサービスを提供するアプリケーションの研究が盛んである．測位手法や行動認識手法を提案するには，開発する際にアルゴリズムが想定通りに稼働しているかの検証やプログラムのデバッグが必要である．しかし，現状の検証やデバッグ方法では視認性に優れないなどの問題があり，開発者の負担となる．そのため，これらのプログラムをデバッグする時に，ログの変化をリアルタイムに可視化してアルゴリズムの検証を容易にする機能が必要である．本研究では，開発者の負担を軽減するために，プログラム内の変数を元にブラウザ側でグラフや地図によるリアルタイムな可視化が可能な開発環境を構築した．システム開発者に構築した開発環境を利用してもらった結果，作業時間が約 64%短縮され，その有用性を確認した．

DAISUKE SAKAMOTO[1] SHOGO HARASAKI[2] KAZUYA MURAO[2] MASAHIRO MOCHIZUKI[3]
NOBUHIKO NISHIO[2]

1. はじめに

スマートフォンの高機能化が進み，さまざまなセンサが搭載されるようになった．加速度，角速度，気圧値などを計測できるセンサデバイスから得られる情報，観測している Wi-Fi 基地局の情報，全天球測位システム（Global Positioning System: GPS）による位置情報などが取得可能である．ひらけた屋外では GPS 測位が用いられるが，屋内や高層ビルが立ち並ぶエリアでは GPS の電波観測が困難であり，GPS 測位は難しい．測位に関する分野では，複数の情報を併用することで，測位精度を向上させる研究や，屋内外で即手法を切り替えてシームレスに測位する研究 [1] が行われている．屋内では，観測された Wi-Fi や Bluetooth low energy (BLE) の電波を利用した測位 [2] や端末のセンサを利用した歩行者自律航法（Pedestrian Dead Reckoning: PDR）による測位 [3] などを利用することで GPS が利用困難なエリアでも測位を可能にしている．

シームレス測位のような複数の測位手法を併用するシステムの開発では，複数のセンサや変数を利用する．システムの動作を検証する際は，開発者が変数の値の意味や特徴を直観的に把握できるように，テキストログよりもグラフや地図などでグラフィカルな可視化をするほうが望ましい．しかし，現状の開発環境でグラフィカルな可視化を行うには，可視化したい変数をログファイルに出力するためのコードをシステムに記載し，出力されたログファイルを表計算ソフトなどで開いてグラフなどを作成する必要がある．システム開発者はシステムが想定どおり動作するまでこの作業を繰り返すため，開発に時間を要する．

そこで本研究では，実行中のスマートフォンアプリが内部でもつ変数などをリアルタイムに可視化することでシステム開発の効率化を図る開発環境を構築する．提案する開発環境はスマートフォンアプリ内に実装したモジュールで，アルゴリズムの計算途中の値や変数を Web ブラウザに転送することで，スマートフォン上でアルゴリズムを実行させながらリアルタイムにブラウザ上でグラフや地図への可視化ができる．加えて，ブラウザ上でグラフ化させた

[1] 立命館大学大学院 情報理工学研究科
Graduate School of Information Science and Engineering, Ritsumeikan University
[2] 立命館大学 情報理工学部
College of Information Science and Engineering, Ritsumeikan University
[3] 立命館大学 総合科学技術研究機構
Research Organization of Science and Engineering, Ritsumeikan University

い変数をクリックすることで，グラフを追加できるため，システム開発者は手軽にスマートフォンアプリ内のデータを可視化できる．

2. 関連研究・技術

2.1 Web ブラウザベースの統合開発環境

Cloud9[*1]は，ブラウザベースの統合開発環境サービスである．このサービスは，ブラウザ上でソースコードを編集できる機能やデバッグ機能を提供している．また，開発したシステムのホスティングサービスを行っており，システム開発者はログインするだけで，すぐにシステムの開発やデバッグを行うことができる．さらに，ブラウザ上で動作するため，オペレーティングシステムや開発マシンの性能に依存しないという特徴を持つ．

2.2 センシングデータ処理ツール

河口ら [4] はセンシングデータ処理ツールの HASC Tool[*2]を開発している．HASC Tool は，スマートフォンアプリケーションの HASC Logger[*3]で収集したセンサデータを元に，グラフによる可視化，信号処理，ラベル付け，HASC Logger と連携した UDP によるセンサデータのリアルタイム転送が可能なツールである．

2.3 検証・デバッグ時の課題

さまざまな測位手法を組み合わせる研究や，ユーザの行動認識などの複数のセンサを使用する研究が提案されている．これらの研究では，開発者の想定通りにシステムが稼働していない場合，原因を探るためデバッグを行う．このデバッグを効率的にするために，利用するセンサデータやプログラム内の変数をグラフや地図を用いて可視化している．現状では，解析したいログを出力できるようにプログラムを書き換え，出力されたログをもとに gnuplot[*4]や表計算ソフトを用いたグラフ化や Google Maps[*5]を用いて地図へのマッピングなどを行い，可視化し検証しなければならない．これらは手作業で行われ，検証ごとに繰り返すのは開発者にとって負担である．一方，2.2 項で説明した HASC Tool を利用すれば，センサデータをツールに転送しグラフによる可視化が可能だが，多様なグラフや地図などによる可視化できないため，可視化としては不十分であることが多い．

また，測位アルゴリズムの検証時は，PC 上のシミュレーションだけでなく，スマートフォンで実行しながらリアルタイムに検証することも必要である．方法として，スマートフォン上にグラフや地図を用意し，センサなどの情報を可視化することが可能である．しかし，スマートフォンの画面領域は狭く複数のグラフを同時かつリアルタイムに描画するのは向いていない．

[*1] https://c9.io
[*2] http://hasc.jp/hc2012/hasctool.html
[*3] http://hasc.jp/hc2012/hasclogger.html
[*4] http://www.gnuplot.info
[*5] https://maps.google.co.jp

3. 要件

3.1 可視化による効率的なデバッグ

既存のデバッグツールを用いたデバッグでは，ログや測位結果が十分に可視化されていないため視認性が低く，デバッグ効率が悪い．よって，グラフや地図を利用した可視化を行うことで，効率的にログの可視化を行えるようにする必要があり，これを実装し，効率の良いデバッグを可能にする．

3.2 可視化方法を追加可能な設計

システム開発者は必要な状況に応じてさまざまな可視化方法を必要とするが，可視化方法を最初から全てを用意するのは難しい．そのためシステム開発者が必要に応じて，可視化方法を追加できるような設計であることが望ましい．そこで，スマートフォン内部のセンサデータやアルゴリズムの計算過程のデータ収集を行う部分と可視化部分を分離することで，新たな可視化方法の追加要求があった際に，可視化部分の更新だけで追加ができる設計にする．

3.3 コードの追加を必要としないログ可視化

アルゴリズムの計算途中の値や変数をグラフによる可視化を行うために，その都度プログラムの修正を行うことは，システム開発者にとって手間である．この手間をなくし，効率的なシステム開発を行うために，システム開発者がソースコードを編集なしに変数などを可視化できる必要がある．変数などを自動的にまとめて収集する仕組みを実装することで，開発者がソースコードにログの収集機能を追加しなくても，変数などを可視化できるようにする．

3.4 測位アルゴリズムのリアルタイム検証

収集したログをもとにグラフを作成する作業では，ログ収集，ログデータの整理，グラフ作成ツールでグラフ作成を繰り返し行うこととなるが，この作業を繰り返し行うのは手間がかかる．そこで，アルゴリズムをスマートフォン上で実行させながら，リアルタイムにグラフ作成を可能にすることで，これらの手間を削減する．

3.5 センサログの記録・再生機能

測位などの検証では，検証場所が遠方などの理由により移動の手間が発生する．また，ログ収集からグラフ作成を行うまでの作業を，遠方の環境で繰り返し行うのは現実的に難しい．そこで，検証場所でセンサログを事前に記録し，記録したセンサログを再生することで，アルゴリズムを現地でなくても検証可能な機能を実装する．これによって，

移動の手間や，後から追加で確認したい変数値やセンサ値があったとしても，いつでも検証が可能となる．

3.6 グラフィカルで操作しやすい画面

グラフによる可視化を行う際には，システム開発者にとって使いやすいようにグラフィカルな画面操作を行えるようになることが望ましい．加えて，スマートフォンの画面領域は狭く，複数のグラフを描画するには難があるため，広い領域での利用ができることが望ましい．これに対して，Web ブラウザを利用して可視化を行うようにし，PC のようなディスプレイ領域が広い環境でも閲覧可能にすることで対処する．これにより，同時に複数のグラフや地図を見ながらのアルゴリズム検証が可能になりデバッグ効率の向上が見込める．さらに，グラフの自由配置が可能となるように実装することで，開発者が着目したいグラフを見やすくできるようにしたり，グラフの比較を行いやすくする．

4. 開発環境の構築

4.1 実装環境

本研究では，測位やセンシングの検証を行うスマートフォン端末として Android[*6] を搭載した端末を，アプリケーションとして我々の研究室が開発している屋内外での測位を目的とした測位エンジンを想定環境として，要件を満たす開発環境の構築を行った．スマートフォンと Web ブラウザ間の通信には Websocket[5] を利用し，通信の中継を行うサーバを Ruby 2.3.1[*7] で実装した．ブラウザ側でのユーザインタフェースの実装に React15.0.2[*8] を利用した．また，グラフの描画には rd3 0.7.4 [*9] を，地図の描画には Google Maps JavaScript API[*10] を利用した．

4.2 開発環境の概要

構築した開発環境の概要図を図 1 に示す．構築した開発環境では，測位の検証を行う Android 端末側にデータ送受信モジュールを実装する．このモジュールはブラウザ側の Viewer との通信，測位エンジン内部のオブジェクトについての収集を行う機能を実装している．開発環境の利用者は，Android 端末上のデータ送受信モジュールを実装した測位アルゴリズムを開始する．起動後，データ送受信モジュールは自動的にインターネット上に配置された中継サーバに対して Websocket の接続を行う．ブラウザと Android 端末間のデータは JSON 形式で通信される．

*6 https://www.android.com
*7 https://www.ruby-lang.org/ja/
*8 https://facebook.github.io/react/
*9 https://github.com/yang-wei/rd3
*10 https://developers.google.com/maps/documentation/javascript/?hl=ja

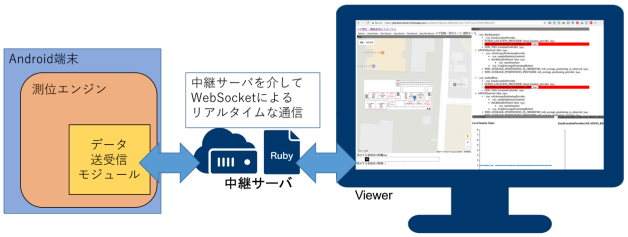

図 1 開発環境の概要図

4.3 Android 端末側の実装

4.3.1 Android 端末からのログ転送

測位アルゴリズムが実装されたクラスの変数の値を収集するため，Java のリフレクションを用いてクラスフィールドを走査し，値の収集を行う．リフレクションとは，実行中のプログラムのクラス名や変数名，パラメータなどを読み取ることができる技術である．クラスフィールドの収集は測位結果が算出されるたびに行われ，収集後，Websocket を通じて中継サーバに対して送信される．転送されたログとデータを元にブラウザ側でグラフ，地図の可視化が行われる．

4.3.2 Android 端末へのコマンド送信

ブラウザ側からコマンドを発行すると中継サーバを介して Android 端末側に転送される．Android 端末側のデータ送受信モジュールで受信したコマンドはモジュール側で解釈を行った上で，測位エンジン側に対して必要な操作を行う．今回はセンサログ記録，記録したログの再生を行うコマンドを実装した．センサログの記録・再生機能は梶ら [6] のものを利用した．センサログ記録コマンドを発行した際には，センサより記録されたデータを Android 端末内部に保持するようになる．センサログ再生コマンド発行時には，記録したセンサログを利用してアルゴリズムの動作検証が可能である．このとき，通常時と同様に記録されたセンサログを Android 端末からブラウザに送信し，ブラウザ側で可視化する．

4.4 ブラウザ側の実装

4.4.1 クラスフィールドのグラフ化

Android 端末で収集され転送されてきたクラスフィールド情報をもとに，ブラウザ側の Viewer にクラスフィールド変数リスト（図 2 の右）が描画される．このリスト内のオブジェクトはそれぞれグラフ化が可能である．リスト中の Open を クリックするとそのオブジェクトがグラフ化（図 2 の左）される．グラフ中の Remove をクリックすることでグラフを消すことが可能である．

4.4.2 地図を利用した可視化

図 3 に地図を利用した可視化の様子を示す．地図には 2 つの機能が実装されている．一つは現在推定位置の表示である．現在推定位置は図中の赤色のピンによって表現され

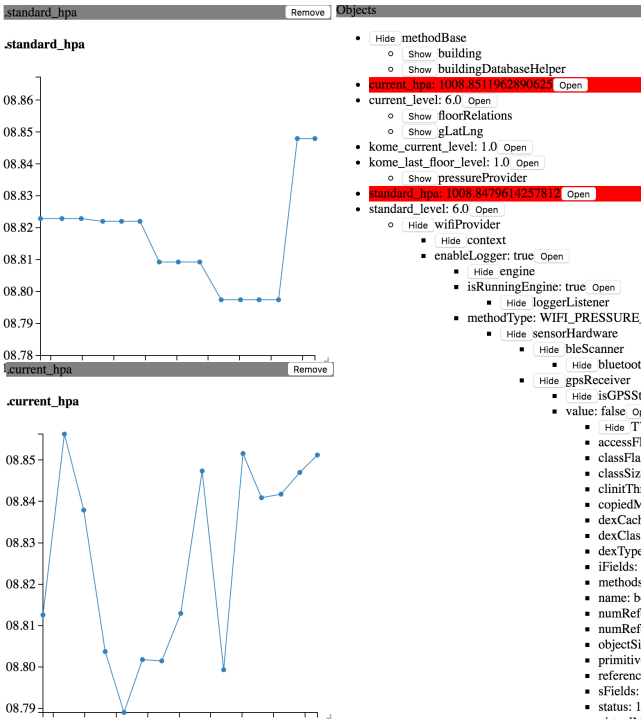

図 2　追加したグラフ（左）とクラスフィールド変数リスト（右）

図 3　地図を利用した可視化

図 4　ログの記録・再生コマンド

る．もう一つの機能は，Wi-Fi 観測情報の可視化である．事前に登録された測位に用いる Wi-Fi 基地局情報と現在観測している Wi-Fi 基地局情報を統合し地図に表示している．図中の青色のピンが事前登録された Wi-Fi 基地局情報，緑色のピンが事前登録された Wi-Fi 基地局で現在 Android 端末上で観測されている Wi-Fi 基地局である．ピンをクリックすることで，BSSID，ESSID，電波強度を表示する．地図下のスライダーでは現在の推定位置を中心とする Wi-Fi 基地局までの距離に応じた表示・非表示の閾値を変更することができる．この機能は Wi-Fi 基地局が大量に登録されている状況下において現在推定位置から遠い位置に登録された Wi-Fi 基地局を表示しなくても良いようにするための機能である．閾値よりも遠い位置にある Wi-Fi 基地局であっても観測されている場合には緑のピンとして描画される．この機能により，どの Wi-Fi 基地局によって測位されているのか，登録されている Wi-Fi 基地局情報に不備がないかを調べることができる．一番下のテキストボックスでは，Wi-Fi 基地局を表示する階層の設定を行う．これは異なる階層の Wi-Fi 基地局のピンを同時に配置した場合に階層の区別ができなくなるのを回避するためである．

4.4.3　可視化方法の追加機能

今回はよく使われる可視化方法として折れ線グラフと地図を標準で実装している．これ以外の可視化手法に関しては，ブラウザ側の実装に対して可視化を行うクラスを作成し，作成したクラスを Viewer に追加実装することで追加が可能である．単純なグラフであれば，可視化に必要なデータ処理，可視化処理を 20 行程度のコードを実装すれば良い．

4.4.4　センサログの記録・再生機能

センサログの記録・再生コマンドの発行は画面上部に設置されたボタンで行う．図 4 にその様子を示す．Start Record をクリックした際は，Android 端末側にログの記録コマンドが発行され，Android 端末側でログの記録が開始される．Stop Record をクリックした際は，ログの記録が終了する．Play Record をクリックした際は，記録されたログが再生され，再生されたログをもとに通常時と同じように可視化される．Stop Play Record をクリックすることでログの再生を中断する．

4.4.5　描画領域の自由配置

地図，グラフ，クラスフィールド変数リストの描画領域はそれぞれユーザが自由に配置することができる．それぞれのパネルの上部をドラッグすることで移動が可能である．図 5 に移動前，図 6 にはグラフを出現させ，地図やクラスフィールド変数リストを入れ替えた後の様子を示す．グラフの配置情報はブラウザ側に保存されているため，ページの再読込を行っても描画領域の配置は維持される．また，グラフは右下部分をドラッグすることで表示領域の拡大縮小が可能である．

5.　評価

5.1　評価環境

測位エンジンの開発経験がある開発者 1 名を被験者とし，評価を行なった．Viewer は，開発者が普段から利用し

図 5　描画領域の移動前・グラフ追加前

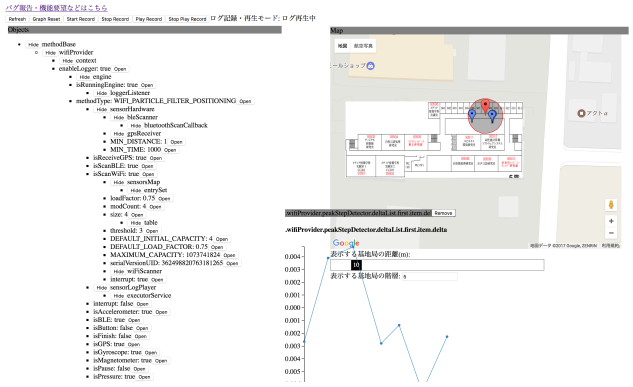

図 6　描画領域の移動後・グラフ追加後

ているノート PC を利用する，中継サーバは，開発者が利用するノート PC とは別のノート PC を利用する．

5.2　定性評価

構築した開発環境を利用した開発者（利用者）の感想や意見，また 3 章の要件を満たしているかを定性評価で示す．

5.2.1　グラフや地図を用いた効率的な可視化

グラフと地図によるリアルタイムな可視化が行え，効率の良いデバッグが可能となったため．3.1 項の要件を満たせた．

5.2.2　可視化方法の追加機能

ブラウザ側に可視化を行うクラスを実装することで，新たなグラフなどの可視化方法を追加することが容易に行えた．棒グラフのような単純なグラフであれば， 20 行程度の Javascript のコードを実装することで，表示可能になった．これにより利用者の要求に合わせて必要な可視化方法が容易に追加可能であると言え，3.2 項の要件を満たせた．

5.2.3　コードの追加を必要としないログ可視化

データ送受信モジュール内部で Java のリフレクション機能を用いることで，動的にシステム実行時の挙動を収集できるようになった．リフレクション機能では，実行中のプログラムのクラス名や変数が読み取ることができる．これにより，別のクラスとして実装されたアルゴリズムであっても，クラスフィールド変数収集部分を変更せずに変数などの収集が可能となり，3.3 項の要件を満たせた．また，利用者からは「コードを追加せずに変数の値がリアルタイムに見ることができるようになり，非常に有用であった」という意見が得られた．

5.2.4　アルゴリズムを実行させながらの情報収集

Websocket を利用してリアルタイムにログをやり取りすることで，アルゴリズム実行時のデータ可視化を可能にし，3.4 項の要件を満たせた．利用者からは「今まではデバッグログを CSV などでテキストとして出力させるためのコードや，画面上に値を表示させるためのコード追加と後々のコード除去に手間がかかっていた上，グラフの作成に Excel などを用いる手間も大きかったが，システムを使うことで実装・検証の際の可視化を削減することができた」という意見が得られた．加えて要望として「詳細に値が見られたり，表示範囲が指定できるなどの高度なプロットができるとよい」という意見が得られた．

5.2.5　センサログの記録・再生機能

センサログの記録・再生機能を実装することで，同一のセンサログによる別のアルゴリズムの検証が可能となった．これにより，センサログを記録した場所と全く同じ状況を再現しながらログの可視化できるようになり，アルゴリズムを変更するたびに現地に行って検証する手間が削減され，3.5 項の要件を満たせた．利用者からは「これまで，実際に実環境でアプリを動かして検証するという方法しか行えなかったため，センサログの記録・再生が簡単に行えるようになり，有用であった.」という意見が得られた．

5.2.6　Web ブラウザを利用したデバッグ環境

ブラウザを利用することで，エクセルのようなアプリケーションを用意しなくても，グラフの描画が可能となった．また，利用者はサイトに[*11]にアクセスするだけで利用ができるようになったため，手軽に開発環境を利用できるようになった．また，複数のディスプレイ環境で構築した開発環境を利用することで，従来のデバッグよりも効率的なデバッグを行うことができると考えられる．さらに，スマートフォンのブラウザからの利用も可能なため，現地でも簡易的にアルゴリズムの挙動確認などが行えるようになった．以上から，3.6 項の要件を満たせた．また，グラフを自由に配置できることに対して，利用者からは「デバッグするにあたって表示させたいグラフを複数個同時に並べることができて有用だった．過去のグラフ配置が保存されるようになっていたため，再度グラフを並べる手間もなかった」という意見が得られた．

5.3　定量評価

定量評価では，システムの検証時間を本研究で構築した開発環境を利用した場合と利用しなかった場合で比較する．気圧センサの値を取得するプログラムの作成を行い，30 秒間の気圧センサの値を収集し，その変化をグラフで可視化

[*11] http://glocationviewer.herokuapp.com/

するまでの時間を評価した．構築した開発環境を利用しなかった場合，気圧センサをログとして出力するソースコードの追加に 31.4 秒，ビルド時間に 10.2 秒，気圧センサのログ収集に 30 秒，スマートフォン端末内に出力された CSV 形式のファイルを PC に転送するのに 13.5 秒，Excel を利用してグラフ化に 14 秒と，合計 99.1 秒かかった．このとき，コード編集，ビルドは一回の時間，それ以降の動作は 3 回行った平均時間である．一方，構築した開発環境を利用した場合，気圧センサをグラフ化するように設定するのに 5.4 秒，気圧センサのログ収集に 30 秒，グラフ化は自動で行われるため 0 秒と，合計 35.4 秒かかった．本研究で構築した開発環境を用いることで，63.7 秒（約 64%）の時間短縮になった．これは構築した開発環境によって，コードの追加，ビルド時間，ログの転送，グラフ化の作業をする必要がなくなったことにより，作業量を根本的に減らすことができたためである．今回の評価では，ログ出力のコード追加が一回で問題なく動作する状況を想定している．そのため，実際には追加したコードそのもののデバッグ時間もかかることから，さらに大きな時間差が現れると考えられる．

6. まとめ

本研究では，スマートフォン搭載の複数のセンサを組み合わせた研究や，その開発時に行われるデータ解析やデバッグの負担を減らすことを目的として，開発環境の設計，実装，評価を行った．また，既存のデバッグ環境では地図のような多様なデータを可視化できない，検証中にリアルタイムに可視化できないという問題を述べた．実行時の変数などを自動収集し，センサデータや計算途中の値をグラフ化することで，デバッグ対象のシステムに対してコードを追加することなく，システム内部の挙動を可視化する開発環境を実現した．また，センサログの記録・再生機能を実装したことで，アルゴリズムを変更するたび現地で検証するという手間をなくした．定性評価では，利用者から「コードを追加せずに変数の値がリアルタイムに見ることができて，便利である」などの良い評価が得られた．定量評価では，構築した開発環境による作業効率の違いを評価し，約 64%の時間短縮となり，その有用性を確認した．

今後として，現在可視化を行っているブラウザ上でアルゴリズムのパラメータを変更し，スマートフォン側に適用する機能の実装が挙げられる．現状では，可視化によって得られた知見を元にして，パラメータを変更する場合には，アルゴリズムを修正し，再度ビルドする必要があり，これを実装することでその手間を省けると考えられる．

参考文献

[1] X. Liu, Q. Man, H. Lu, and X. Lin.Wi-fi/marg/gps integrated system for seamless mobile positioning. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 2323–2328, April 2013.

[2] J. Hightower and G. Borriello. Particle filters for location estimation in ubiquitous computing: A case study. In Nigel Davies, Elizabeth Mynatt, and Itiro Siio, editors,*Proceedings of the Sixth International Conference on Ubiquitous Computing (Ubicomp 2004)*, Vol. 3205 of Lecture Notes in Computer Science, pp. 88–106. Springer-Verlag, September 2004.

[3] S. Yoshimi, K. Kanagu, M. Mochizuki, K. Murao, and N. Nishio. Pdr trajectory estimation using pedestrian-space constraints: Real world evaluations. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, UbiComp/ISWC'15 Adjunct, pp. 1499–1508, New York, NY, USA, 2015. ACM.

[4] N. Kawaguchi, N. Ogawa, Y. Iwasaki, and K. Kaji. Distributed human activity data processing using hasc tool. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, pp. 603–604, New York, NY, USA, 2011. ACM.

[5] I. Fette and A. Melnikov. The WebSocket Protocol. RFC 6455 (Proposed Standard), December 2011. Updated by RFC 7936.

[6] K. Kaji, K. Kanagu, K. Murao, N. Nishio, K. Urano, H. Iida, and N. Kawaguchi. Multi-algorithm on-site evaluation system for pdr challenge. In *2016 Ninth International Conference on Mobile Computing and Ubiquitous Networking (ICMU)*, pp. 1–6, Oct 2016.