情報処理学会論文誌 Vol.59 No.12 2221–2231 (Dec. 2018)

推薦論文

LeafletとOpenStreetMapを用いたFocus+Glue+Contextマップインタフェースの開発と評価

竹内 健祐[1,a)] 山本 大介[1,b)] 高橋 直久[1,c)]

受付日 2018年3月1日, 採録日 2018年9月7日

概要：地図において注目地点を拡大し周辺を縮小して視認性と一覧性を向上させる描画方式に関する研究は広く行われており，我々も注目する地点を拡大表示するFocus領域，その周辺の地域を表示するContext領域，2つの領域間の縮尺の違いによる歪みを吸収するGlue領域の3つからなるFocus+Glue+Contextマップを用いたWebマップシステムEMMAを提案してきた．しかし，場所によりスケールの異なる地図においては，複数のレイヤに分けて表示することができず，地図オブジェクトの描画・移動に関しては位置を計算して配置する必要があった．上記の問題を解決するために本研究ではEMMAを発展させて，構造化した2つのLeafletを有し，Focus・Glue・Contextの異なる座標系のマルチレイヤ構造を持つWebマップシステムを提案する．また，2つのLeaflet間を連携させて整合性を保つ機能，および，地図オブジェクトの描画・移動操作を監視して動的に位置補正する機能を実現する．これにより，可変スケールマップをベースにした位置情報サービスの実現を容易にする．また本論文では提案システムに基づくプロトタイプシステムを実装した．さらに，作成したプロトタイプシステムを用いた実験的評価により，提案システムの有用性を明らかにした．

キーワード：地理情報システム，Webマップシステム，OpenStreetMap，モバイルアプリケーション，ナビゲーションシステム

Development and Evaluation of Focus+Glue+Context Map Interface Using Leaflet and OpenStreetMap

Kensuke Takeuchi[1,a)] Daisuke Yamamoto[1,b)] Naohisa Takahashi[1,c)]

Received: March 1, 2018, Accepted: September 7, 2018

Abstract: In the field of maps, research on a drawing method that improves visibility and listiness by expanding the point of interest and reducing the surroundings has been widely conducted. Web map services will become easy to handle by using open map data like OpenStreetMap and open-source libraries for interactive maps like Leaflet. We have already proposed and implemented Focus+Glue+Context map named EMMA for Web map services. The Focus+Glue+Context map is composed of a Focus area for enlarging the point of interest, a Context area for displaying the surrounding area, and a Glue area for absorbing distortion due to the difference in scale between the two areas. However, maps with different scales depending on places can not be divided into a plurality of layers and can not be displayed separately. In order to move a map object, we need to calculate the position. To solve the above problem, in this research, we propose a web map system with two structured leaflets and a multi-layer structure with different coordinate system. This makes it easy to realize the location information service based on the variable scale map. In this paper, we have implemented a prototype system based on the proposed system. Furthermore, we have clarified the usefulness of the proposed system by experimental evaluations using the prototype system.

Keywords: geographic information system, Web map system, OpenStreetMap, mobile application, navigation system

[1] 名古屋工業大学大学院工学研究科
Graduate School of Engineering, Nagoya Institute of Technology, Nagoya, Aichi 466–8555, Japan
[a)] takeuchi@moss.elcom.nitech.ac.jp
[b)] yamamoto.daisuke@nitech.ac.jp
[c)] naohisa@nitech.ac.jp

本論文の内容は2017年6月のマルチメディア，分散，協調とモバイル（DICOMO2017）シンポジウムにて報告され，高度交通システムとスマートコミュニティ研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

1. はじめに

注目地点を拡大し周辺を縮小して視認性と一覧性を向上させる描画方式に関して古くから種々の研究がなされている．Furnas の魚眼レンズのメタファによる描画方式 [1] は，これらの研究の起点となる研究であり，魚眼マップ（あるいは Focus+Context マップ）として地理情報の描画にも採り入れられて発展している．Harrie ら [2] は場所により縮尺が異なる可変スケールマップの実現機能を提案し，Craig ら [3] はモバイル端末において画面スペースを有効に使うことのできる魚眼マップを実現した．また，我々は，Focus+Glue+Context マップを提案し，伸縮自在のマップシステム EMMA（Elastic Mobile Map System）の研究を進めてきた [4], [5], [6], [7], [8]．Focus+Glue+Context マップは，**図 1** に示すように，注目する特定の地域を拡大して表示する Focus 領域，周辺の地域を表示する Context 領域，スケールの異なる 2 つの領域間の縮尺率の違いによって生じる歪みを吸収する役割を果たす Glue 領域からなり，視認性と一覧性を両立させた Web マップを目指している．

また，OpenStreetMap [9] に代表されるオープンデータ形式の地図データや，Leaflet [10] や OpenLayers [11] に代表されるオープンソース型の地図制御用ライブラリ（以下，地図制御ライブラリと呼ぶ）の登場により，Web マップサービスは扱いやすいものとなった．地図制御ライブラリは，タイリング技術により早い応答速度で地図を描画することが可能であると同時に，マーカやポリゴンなどの地図オブジェクトを表示したり，多数のレイヤを重ね合わせて表示することができる．これらの特徴を生かし，文献 [12] のように，地図の情報量を増やすために他のサービスや情報をマッシュアップし，レイヤとして重ね合わせて表示することが可能になった．

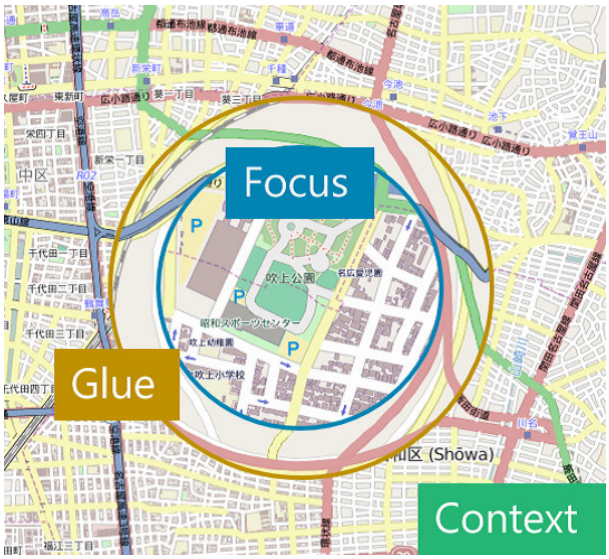


図 1 Focus+Glue+Context マップ
Fig. 1 Focus+Glue+Context map.

一方で，既存の地図制御ライブラリは，場所により縮尺の変わらない均一スケール座標系での利用を前提としており，魚眼マップに代表されるような可変スケール座標系には適用困難であった．我々は，Focus+Glue+Context マップを Leaflet を用いて開発してきた [8] が，応答速度や，地図オブジェクトの描画に課題があり，地図制御ライブラリの特徴を生かしきれなかった．

本研究の目的は，Leaflet に代表される地図制御ライブラリを，可変スケール座標系の地図である Focus+Glue+Context マップに適用可能にする仕組みを提案することである．これにより，均一スケール座標系の Leaflet と同様な操作・開発法で，Focus+Glue+Context マップを制御可能になる．これらを実現するうえでの問題を以下に述べる．

問題点 1 既存の地図制御ライブラリ [10], [11] は均一スケール座標系に対して，タイリング技術を適用することが前提のシステムである．そのため，既存の Focus+Glue+Context マップ [8] は Context のみタイリング技術を用いて表示していたが，Focus や Glue に適用できず，応答速度に課題があった．

問題点 2 既存の地図制御ライブラリは均一スケール座標系に対して，地図操作を可能にすることが前提のシステムである．そのため，Focus や Glue などの，複数の異なる座標系を持つ地図を正しい位置で重ね合わせて操作可能にすることができなかった．

問題点 3 既存の地図制御ライブラリは均一スケール座標系に対して，マーカやポリラインなどの地図オブジェクトを表示することが前提のシステムである．Focus+Glue+Context マップのような可変スケール座標系では，マーカやポリラインなどの地図オブジェクトを正しい位置に描画することができなかった．

これらの問題を解決するために，2 つの異なる座標系を持つ Leaflet を連携させる仕組みを実現することにより，Focus+Glue+Context の異なる座標系のマルチレイヤ構造を持つ Web マップシステム「OpenEMMA」を提案する．ここで，第 1 の Leaflet は，均一スケール座標系で全画面を描画し，第 2 の Leaflet は，可変スケール座標系である Focus+Glue を描画する．これにより，Focus+Glue 領域に対してもタイリング技術を適用することで，応答速度の改善を可能にする．さらに，2 つの Leaflet 間を連携させて位置的な整合性を保つ機能，および，地図オブジェクトの描画・移動操作を監視して動的に位置補正する機能を実現する．これにより，Focus+Glue+Context マップの開発者や利用者が，これらの座標系の違いを意識することなく，地図や地図オブジェクトの描画や操作が可能になる．

2. 関連研究

一般に，人が頭の中に持つ地理的イメージ（認知地図 [13], [14]）は，実際の地形と必ずしも一致せず，歪んで

いることも多い．そこで，認知地図と実際の地形の対応関係の形成を助けるために，地図を歪めて表示させる研究は広く行われてきた [15], [16]．

たとえば，全体的な概要と詳細情報を 1 つの画面で表示することを目的とした Focus+Context 型マップに関する研究がある．Furnas [1] は，Focus からの距離に応じて周囲の情報の詳細度を変化させる魚眼レンズ状の情報表示手法について提案した．また，Sarkar ら [17], [18] は，Focus+Context 型マップの概念を地理的な地図や図形に適用した．これらの論文では，単一の再配置関数を用いて，まるで魚眼レンズを当てたかのように地図全体を変形させる手法について述べている．また，Harrie ら [2] は，再配置関数を用いて，Focus 領域を歪みなく拡大して表示する代わりに，Context 全体に歪みを持たせて描画する手法について提案した．Craig ら [3] は，モバイル端末においての魚眼レンズマップの有用性を評価・提案している．Wang ら [20] は地下鉄マップに Focus+Context マップを適用している．文献 [1], [2], [3], [17], [18], [20] の可変スケール座標系マップの研究では，地図全体が歪んでいるため，Lealfet などの地図制御ライブラリを適用できず，画面全体を 1 枚の地図として描画している．

また，Takahashi [4] は，再配置関数をベースにした地図描画に対して，歪みを吸収する Glue 領域を導入した Focus+Glue+Context 型マップを提案した．拡大で生じるマップの歪みを Glue に集中させることで，Focus と Context を歪みのない均一なスケールで表示できる．Yamamoto ら [5] は，上記のシステムを Web マップサービスとして実現し，Web マップとして十分に許容される範囲の応答時間になることを実証しているが，オープンソース地図技術を用いていないクローズドなシステムであった．さらに，Yamamoto ら [8] では，提案手法と同様に，Leaflet と OpenStreetMap を用いて Focus+Glue+Context 型マップを実現した．しかしながら，1 カ所（Context）のみしか Leaflet で描画しておらず，Focus はサーバから 1 枚画像を取得して表示する方式であったため，問題点 1 から 3 のように Focus の応答速度や使い勝手に問題があった．

また，インタラクティブな地図を作るために，様々な地理的情報を，複数のレイヤに分けて表示することは一般的である．Zaslavsky [19] は，複数のレイヤを用いて XML ベースの空間データを統合した地図ソフトウェアを開発している．また，文献 [12] のように近年，地図の情報量を増やすために他のサービスや情報をマッシュアップし，レイヤとして重ね合わせて表示するものが増えている．Piškorec ら [21] は Web マップと関連付けて，多層ネットワークを描画することができる．しかし，文献 [12], [19], [21] は，ベースとなる Web マップと同じ座標系を持つレイヤを重ね合わせることが前提のシステムであり，可変スケール座標マップなどの異なる座標系のレイヤを重ね合わせて表示することはできない．提案手法は，Focus+Glue+Context の異なる座標系のマルチレイヤ構造を持つと同時に，これらを正しく重ね合わせて表示できるシステムを提案し，可変スケールマップをベースにした位置情報サービスの実現を容易にする．

また，オープンデータとオープンソースを組み合わせた研究は多くある．たとえば，OpenStreetMap [9] に代表されるオープン地図データと，それを扱うオープンソースである Leaflet [10] や OpenLayers [11] を用いた研究 [23], [24] がある．さらに，航空交通 [22] や Linked Open Data [25] など，様々な形式のオープンデータが普及している．オープンデータとオープンソースを組み合わせたシステムを構築することの有用性は，新しい形式のオープンデータへの対応が容易であること，データとセットでシステムを提供することによりオープンデータの普及を促すことができることなどがあげられる．

3. 提案システム

3.1 提案システムの特徴

提案システムの主な特徴は次のとおりである．

EMMA レイヤ

表 1 のように，Focus の Leaflet（F-Leaflet）と Glue の地図画像（ラスタマップ）からなるレイヤ（EMMA レイヤ）を Context の Leaflet（C-Leaflet）に加えて，ネスト構造の Leaflet を構築する．各レイヤとサブレイヤの座標系を表 1 のように定めることにより，場所により縮尺を変えて描画する EMMA 方式に対応させる．EMMA レイヤでは，Focus のラスタマップを Leaflet のタイリング機能により描画し，これに，Glue のラスタマップを合成する．Web ブラウザで，Focus の均一縮尺のラスタマップと Glue の非均一縮尺のラスタマップを個別にサーバから読み込み，合成して描画することにより，サーバの負荷を減らし応答性を向上させる．これにより問題点 1 を解決する．

Leaflet 連携機能

F-Leaflet と C-Leaflet の間に入って各 Leaflet を制御する機能を実現し，縮尺の違いから生じる各領域間での表示の不整合が起きないように，つねに 2 つの Leaflet を監視して表示位置や縮尺などを管理する．これにより問題点 2 を解決する．

地図オブジェクトの動的位置補正機能

縮尺が異なる各領域間で，地図オブジェクトを各領域の縮尺に応じて形状を変更したり表示位置を変更する．また，この機能により地図オブジェクトの位置指定のためのマウスクリックなどにより発生するイベントの処理において，場所による縮尺の変化を考慮せずに描画しても，各領域内の地図オブジェクトの位置の整合性を保つことができる．これにより問題点 3 を解決する．

表 1 OpenEMMA のレイヤ
Table 1 Layers of OpenEMMA.

z-index	レイヤ	Leaflet	pane	サブレイヤ	Leaflet	座標系
200	Context	C-Leaflet	tile	-	-	Context
300	EMMA	C-Leaflet	focusglue	Glue	-	Glue
				Focus	F-Leaflet	Focus
400	道路・ポリゴン	C-Leaflet	overlay	-	-	Context
600	マーカ	C-Leaflet	marker	-	-	Context
700	ポップアップ	C-Leaflet	popup	-	-	Context

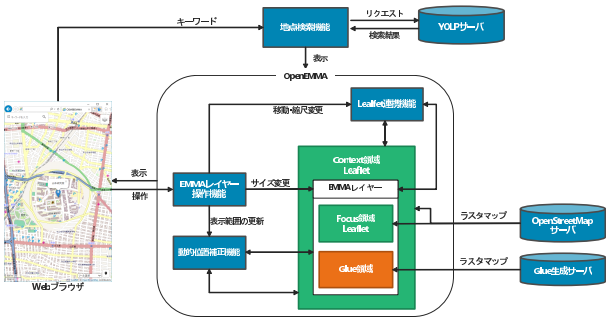


図 2 提案システムの構成図
Fig. 2 The structure of the proposed system.

3.2 システム構成

本論文では下記の特徴を持つ OpenEMMA システムを提案する．提案システムは**図 2** のように Context 領域 Leaflet，EMMA レイヤ，EMMA レイヤ操作機能，Leaflet 連携機能，動的位置補正機能からなり，OpenStreetMap サーバと Glue 生成サーバのラスタマップを用いる．また，地点検索機能と連携して，位置情報サービスを実現する．

YOLP サーバは Yahoo! Open Local Platform [26] のことで，Yahoo!JAPAN がデベロッパ向けに提供する地図・地域情報の API・SDK である．

地図検索機能は，ユーザが入力したキーワードを YOLP サーバに送り，返ってきた JSON ファイルをもとに地図の中心から近い順に検索結果を表示し，Focus+Glue 領域を作成する機能である．ユーザは，この機能を例に，所望の空間データを地図オブジェクトとして Leaflet のレイヤに描画する機能を実現すると，レイヤの重ね合わせにより，独自にカスタマイズした Focus+Glue+Context マップを容易に作成できるようになる．

OpenStreetMap サーバはタイリング用のラスタマップを生成し，Glue 生成サーバは Glue 領域用に道路を間引いて描画したラスタマップ [8] を生成する．

F-Leaflet と C-Leaflet は OpenStreetMap サーバからラスタマップを取得して表示する．Glue は Glue 描画機能を用いて，Glue 生成サーバから道路網を総描したラスタマップ取得して表示する．

EMMA レイヤ操作機能では EMMA レイヤのサイズ変更や移動，縮尺変更を行うことができる．

Leaflet 連携機能と動的位置補正機能はつねに Focus と Context の状態を監視しており，ユーザの操作によって Focus や Context の状態が変化したときに動作する．

OpenEMMA システムは，Focus の生成，移動，サイズ変更のための操作インタフェースを提供する．ユーザは，Web ブラウザを介して，これらの操作機能を利用し，所望の場所と形状の地図情報を表示した Focus+Glue+Context マップを得る．実現法については 4 章で述べる．

4. 実現法

本章では，OpenEMMA システム，および，OpenEMMA システムを用いた地図検索機能の実現法について述べる．

4.1 EMMA レイヤ

EMMA レイヤは Focus と Glue の地図画像を合成して描画するレイヤである．

4.1.1 EMMA レイヤの作成

Focus は Context 上の指定された領域を拡大描画する．このため，Leaflet のイベントハンドラを用いて，以下の手順で，ユーザがクリックにより指定した地点に Focus と Glue からなる EMMA レイヤを作成する．

手順 1 Leaflet のイベントハンドラを用いて，Context において地図上の点をクリックしたときの緯度経度を取得する．

手順 2 手順 1 で取得した緯度経度を Leaflet の latLngToLayerPoint メソッド[*1]を用いて Web ブラウザ上での xy 座標標に変換する．

手順 3 EMMA レイヤの HTML 要素を生成し，その中に Focus と Glue の HTML 要素を生成する．このとき Focus と Glue は css の border-radius を用いて円形になるようにする．

手順 4 手順 2 で得た Web ブラウザ上の xy 座標に Focus と Glue を移動させる．このとき，手順 2 で得た Web ブラウザ上の座標 (x, y) と Focus の半径 r_f から，Focus の座標の原点 (x_f, y_f) を**図 3** のように $(x - r, y - r)$ と定める．Focus 内の点の座標は，原点からの方向と

[*1] 緯度経度が与えられた場合，原点ピクセルを基準として対応するピクセル座標を返すメソッド．

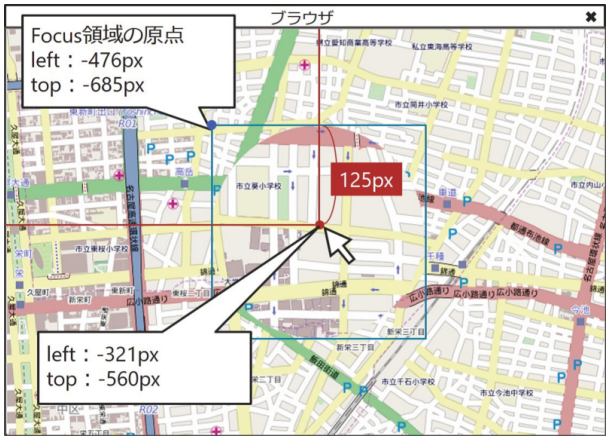


図 3 Focus 領域の原点
Fig. 3 The origin of the Focus-coordinate axes.

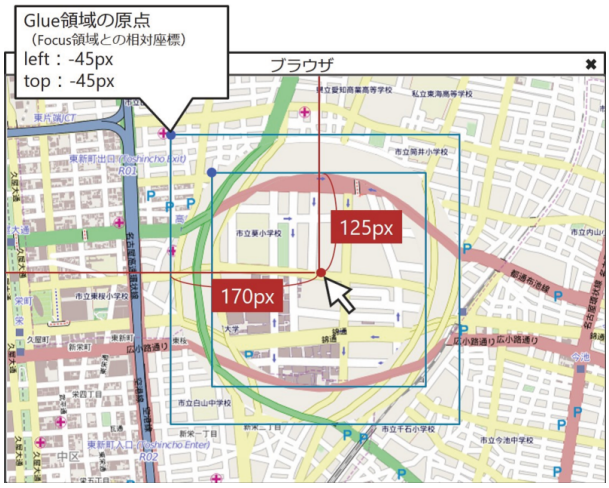


図 4 Glue 領域の原点
Fig. 4 The origin of the Glue-coordinate axes.

距離により定まる．また Focus の座標の原点 (x_f, y_f) から Glue の半径 r_g と Focus の半径 r_f の差 $r_g - r_f$ を引いたものを Glue の Focus からの相対座標とする（図 4）．

手順 5 Focus の HTML 要素に Leaflet インスタンスを作成する．

手順 5 Glue サーバが提供するラスタマップを Glue に表示する．

4.1.2 EMMA レイヤの Leaflet レイヤ化

Leaflet [10] は**表 2** のように pane と呼ばれるレイヤからなる層構造になっており，マップのタイル画像を表示する tilePane や，マーカを表示する markerPane など，複数のレイヤを重ね合わせて描画している．このとき z-index の値が大きいレイヤが上になる．一般にこのレイヤは同じ座標系持ち，同一 Leaflet 内にあるレイヤは，地図の描画範囲が変わると連動して一緒に動く．

EMMA レイヤは場所によって縮尺が異なるため，各領域で座標系が異なる．また，Focus と Glue は一体となって動く必要がある．

表 2 Leaflet の階層
Table 2 Structure of Leaflet.

Pane	Z-index	説明
tilePane	200	マップのタイル画像を表示する層
overlayPane	400	ポリゴンやパスなどを表示する層
shadowPane	500	影を表示する層
markerPane	600	マーカを表示する層
tooltipPane	650	ツールチップを表示する層
popupPane	700	ポップアップを表示する層

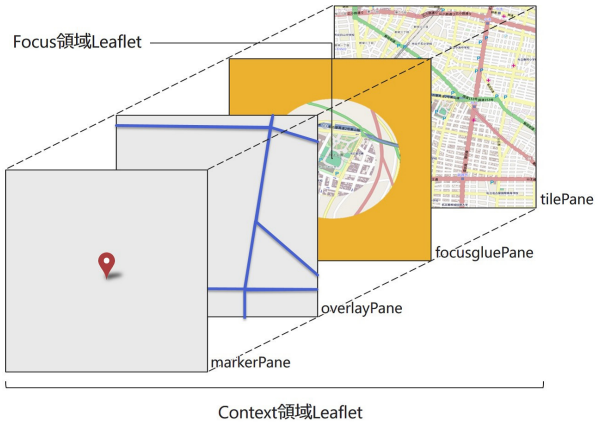


図 5 提案システムでの Leaflet 階層
Fig. 5 Structure of Leaflet in the proposed system.

そこで，異なる座標系のレイヤを連動させて動作させるためにネスト型の Leaflet を構築する．すなわち，C-Leaflet に EMMA レイヤを配置し，そのサブレイヤとして F-Leaflet と Glue を配置する（表 1）．これによって Focus と Glue は Context の動きに連動して一緒に動くようになる．また Focus と Glue の上にポリゴンやマーカなど，C-Leaflet の他のレイヤを重ねて表示できるようになる（**図 5**）．

4.2 EMMA レイヤ操作機能

ユーザはドラッグやホイール操作で EMMA レイヤを操作する．本節ではサイズ変更操作について述べ，移動と縮尺変更の操作については 4.3 節で述べる．

ユーザは，Focus と Glue の特定の場所に対するドラッグ操作により，この領域のサイズ変更要求を指示する．以後 Focus と Glue を一体として扱う場合，この 2 つの領域を Focus+Glue 領域と呼ぶ．EMMA レイヤは，次の手順で，Focus+Glue 領域の大きさを変化させ，再描画する．

手順 1 Glue 内にマウスポインタがあるときにドラッグをすると，サイズを変更する処理を開始する．ドラッグした際に C-Leaflet が動かないように，C-Leaflet においてオプションの dragging の値を false に設定する．

手順 2 ドラッグ中はつねにマウスポインタの座標を取得し，Focus の中心座標からの距離を計算する．

手順 3 この距離を一定時間前の Focus+Glue 領域の半径と比較し，差分を各領域のサイズに適用する．

4.3 Leaflet 連携機能

Focus と Context の間に入って各 Leaflet を制御する機能である．

4.3.1 EMMA レイヤの移動

EMMA レイヤの移動は，Focus+Glue 領域の座標移動とそれにともなう領域の再描画の 2 つからなる．EMMA レイヤは，ユーザのドラッグ操作に応じて，以下のように Focus+Glue 領域を移動させる．

ドラッグ開始時 Focus をドラッグした際に C-Leaflet が動かないように，C-Leaflet においてオプションの dragging の値を false に設定する．

ドラッグ中 Focus+Glue 領域を移動させる．

ドラッグ終了時 C-Leaflet においてオプションの dragging の値を true に再設定する．

また，Focus+Glue 領域の座標が変化したときに，Web ブラウザ上での Focus+Glue 領域の中心 xy 座標を Leaflet の layerPointToLatLng メソッド[*2]を用いて緯度経度に変換する．得た緯度経度で Focus+Glue 領域を再描画することにより，マウスポインタの位置の変化に，画面上の Focus+Glue 領域を追従させる．

4.3.2 縮尺変化時の位置変更

Context や Focus の縮尺がユーザの操作によって変化したとき，システムは Focus+Glue 領域を縮尺の変化に合わせて移動させる必要がある．Context と Focus の拡大縮小に対してのイベントリスナを作成し，縮尺が変わるたびに縮尺変更後のブラウザ上での位置を計算して Focus+Glue 領域を再配置する．

4.4 領域間での動的位置補正機能

Focus+Glue+Context マップは場所により縮尺が異なり非均一な座標系であるため，提案システムは，各領域ごとに座標系を定め，地図オブジェクトの所在場所に応じて表示位置と形状を計算して描画する．また，Focus の生成，消滅，移動が発生した場合は画面上の地図オブジェクトの座標を再計算する必要がある．このとき，Context の座標系を基準として用いる．Context は均一の縮尺であるので，地図オブジェクトが Context にある場合は所在場所の緯度経度に応じた座標に描画すればよい．地図オブジェクトが Focus+Glue 領域にある場合は，Context の座標系に変換してから描画する必要がある．Leaflet の地図オブジェクトは地理座標をもとに描画されているため，表示位置や形状を変化させるために，地図オブジェクトの持つ地理座標 (lat, lng) にある点 P を $P' = (lat', lng')$ に変換する．

まず，P が存在する領域を返す関数 $region(P)$ を実現する．

$$region(P) = Focus|Glue|Context|NULL$$

P が存在する領域は，Focus，Glue，Context のいずれかであるか，または画面上に存在しないかである．

次に地理座標 (lat, lng) を，Focus 座標系，Glue 座標系，Context 座標系の位置座標 (x_F, y_F)，(x_G, y_G)，(x_C, y_C) に変換する次の 3 つの関数を実現する．

$$\begin{aligned} f_F(lat, lng) &= (x_F, y_F)(\text{P が Focus にある場合}) \\ &= NULL(\text{そうでない場合}) \\ f_G(lat, lng) &= (x_G, y_G)(\text{P が Glue にある場合}) \\ &= NULL(\text{そうでない場合}) \\ f_C(lat, lng) &= (x_C, y_C)(\text{P が Context にある場合}) \\ &= NULL(\text{そうでない場合}) \end{aligned}$$

また，Focus 座標系，Glue 座標系，Context 座標系の位置座標 (x_F, y_F)，(x_G, y_G)，(x_C, y_C) を地理座標 (lat, lng) に変換する次の 3 つの関数を実現する．

$$\begin{aligned} f_F^{-1}(x_F, y_F) &= (lat, lng)(\text{P が Focus にある場合}) \\ &= NULL(\text{そうでない場合}) \\ f_G^{-1}(x_G, y_G) &= (lat, lng)(\text{P が Glue にある場合}) \\ &= NULL(\text{そうでない場合}) \\ f_C^{-1}(x_C, y_C) &= (lat, lng)(\text{P が Context にある場合}) \\ &= NULL(\text{そうでない場合}) \end{aligned}$$

ここで Focus，Glue，Context の位置座標原点を (x_{FO}, y_{FO})，(x_{GO}, y_{GO})，(x_{CO}, y_{CO}) とし，$(x_{CO}, y_{CO}) = (0, 0)$ である．

Focus 上の場合

$region(P) = Focus$ のとき，P の Focus での位置座標は $(x_F, y_F) = f_F(lat, lng)$ であるので，Context での位置座標は $(x_{FO} + x_F, y_{FO} + y_F)$ となる．これより変換後の地理座標 $P' = (lat', lng')$ は，

$$P' = (lat', lng') = f_c^{-1}(x_{FO} + x_F, y_{FO} + y_F)$$

で求められる．

Glue 上の場合

Glue は Focus–Glue 間と Glue–Context 間を滑らかに接続するために 3 次ベジエ曲線で表現される再配置関数を用いて地図の変形を行っている（**図 6**）．3 次ベジエ曲線で表現される再配置関数を $g(lat, lng) = (x_G, y_G)$ と定義すると，Glue での位置座標は $(x_G, y_G) = g(lat, lng)$ であるので，Context での位置座標は $(x_{GO} + x_G, y_{GO} + y_G)$ となる．これより変換後の地理座標 $P' = (lat', lng')$ は，

$$P' = (lat', lng') = f_c^{-1}(x_{GO} + x_G, y_{GO} + y_G)$$

で求められる．

[*2] 原点ピクセルを基準にしたピクセル座標が与えられた場合，対応する地理座標を返すメソッド．

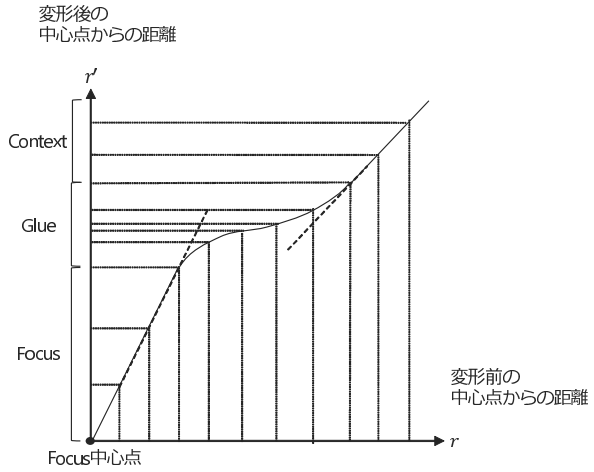


図 6 Focus+Context+Glue マップの再配置関数形状
Fig. 6 The shape of relocation function for Focus+Context+Glue map.

4.5 地点検索機能

地点検索機能は，以下の手順で施設を検索し，結果を描画する．

(1) 地図画像に対して，キーワードと検索件数が入力されると，地図の中心座標（緯度経度）と入力値を YOLP に送り，json 形式の検索結果を受け取る．
(2) 検索結果から施設の名称と座標の組を抽出して施設一覧表を作成する．
(3) Leaflet の markerPane において，施設一覧表の全施設の座標にマーカを表示する．
(4) 施設一覧表の全施設に対して，施設名がクリックされたときに，その施設の座標を中心とした Focus+Glue 領域を描画するように，クリッカブルな施設名リストを作成して表示する．

5. プロトタイプシステム

4 章で述べた全機能とスマートフォン向け UI を有する提案システムのプロトタイプを開発した．開発には，Windows10, Visual Studio Community 2015, JavaScript, HTML, CSS を用いた．動作確認には主に Google Chrome と iOS 端末として iPad Air2 を用いた．

5.1 操作コマンド

プロトタイプを起動すると，**図 7** のような Context のラスタマップ，および，ツール選択（画面右下）と地点検索（画面左上）のプルタウンメニューが表示される．ツール選択メニューからは，可動 Focus 生成，固定 Focus 生成，Focus+Glue 削除の操作コマンドを選択する．地点検索については 5.3 節で述べる．可動 Focus 生成を選択すると，プロトタイプは自由に動かせる Focus+Glue 領域をクリックした場所に作り，描画する（**図 8**）．固定 Focus 生成を選択すると，プロトタイプは，ドラッグ操作をしても移動しない Focus+Glue 領域をクリックした場所に生成し，描画する．Focus+Glue 削除を選択して，画面上の Focus+Glue 領域をクリックすると，プロトタイプは，その領域を削除する．

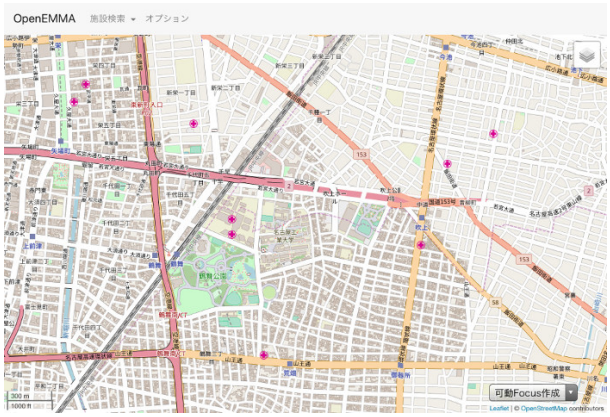


図 7 起動画面 iOS
Fig. 7 Initial screen shot of the prototype system on iOS.

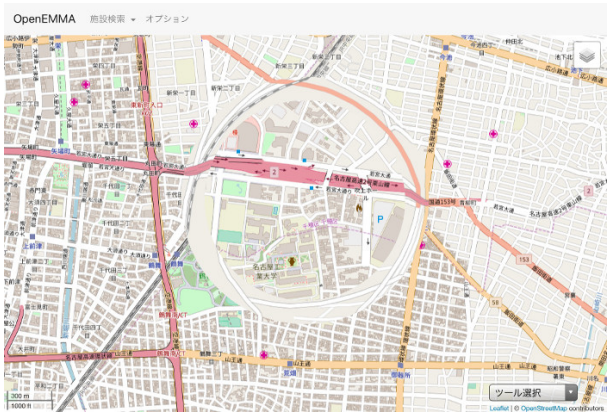


図 8 Focus+Glue 領域の作成
Fig. 8 Creating a Focus+Glue Area.

5.2 Focus+Glue 領域の移動とサイズ変更

Focus+Glue 領域を移動するにはユーザは Focus または Glue 上にマウスを持っていき，ドラッグする．ドラッグ操作の実現には jQuery UI の Draggable [27] を使用した．Draggable は要素をドラッグできるようにする jQuery UI の機能であり，ドラッグ開始時，ドラッグ中，ドラッグ終了時に使うイベントハンドラ関数をあらかじめ指定しておくと，ドラッグ操作イベントの発生時に，その関数が起動される．

Focus+Glue 領域のサイズを変更するにはまずユーザは Glue 上にマウスを持っていく．システムは，マウスが Glue 上にあると領域に影のエフェクトをかけて，ドラッグするとサイズを変更できることを知らせる．スマートフォンでは直接 Glue 上をタッチしてドラッグすることでサイズ変更ができる．デフォルトでは Focus と Glue を同時にサイズ変更するようにしているが，画面上部のオプションによ

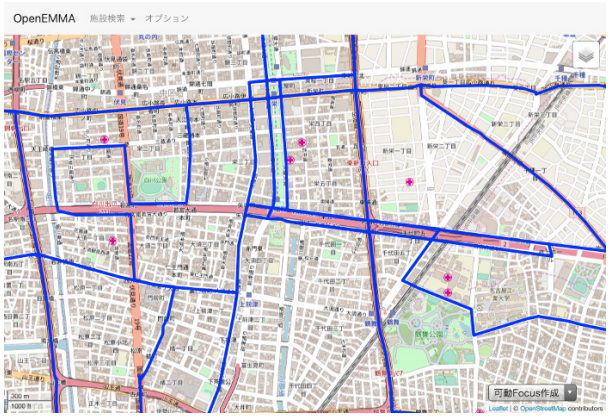


図 9 Leaflet 上のポリライン
Fig. 9 Drawing polylines.

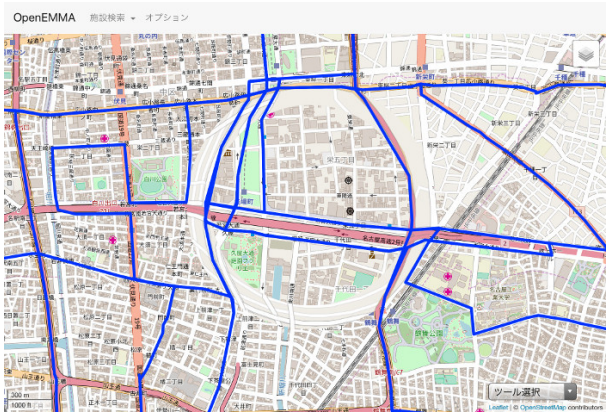


図 10 各領域に応じたポリラインの表示
Fig. 10 Drawing polylines adapted to each area.

り，各領域を個別にサイズ変更するようにも設定する機能も実現している．

5.3 レイヤの重ね合わせ

ユーザは表示したい地図オブジェクトを Leaflet の overlayPane に描画することによって**図 9** のように地図上に描画することができる．このとき，道路が Focus+Glue 領域にある場合にも，プロトタイプは動的位置補正機能により**図 10** のように，位置の整合性を保つように描画する．

また，ユーザが画面左上の施設検索において，調べたいキーワードを入力し検索ボタンを押すと検索結果としてクリッカブルな施設名の一覧が表示される．さらに，一覧から施設名を選択してクリックすると，プロトタイプは，その施設の所在地点に Focus+Glue 領域を作成する．

5.4 タッチ操作対応 UI

スマートフォンでの操作は PC とは異なるために，JavaScript のライブラリである jQuery [27] を用いて PC のマウス操作・スマートフォンのタッチ操作のどちらでも EMMA システムが動くようなプロトタイプのユーザインタフェースを開発した．また，Bootstrap [28] を用いることで，画面幅に合わせてレイアウトを変化させるインタラクティブな UI を開発した．

6. 評価実験

以下の PC とブラウザでプロトタイプを動作させて，提案システムの利便性を実験的に評価した．

製造元 DELL
モデル Precision 3420
プロセッサ Intel(R) Core(TM) i5-6500 CPU @ 3.20 GHz 3.19 GHz
RAM 8.00 GB
OS Windows10
Web ブラウザ chrome

6.1 実験内容

実験 1

(1) 比較対象
- (a) 提案手法．Web ブラウザが Leaflet のタイリング機能により Focus と Glue のラスタマップを合成する．提案システムのプロトタイプを使用．
- (b) 従来手法．サーバが Focus と Glue のラスタマップを合成する．今回は従来手法の実現のために，中心座標・縮尺・サイズをパラメータとして地図画像を返すサーバを作成．

(2) 比較項目
- Focus の平均描画時間（ms）

(3) 測定対象

次の 3 つの大きさ（ピクセル）の Focus に対して，各 30 回測定する．
- 400 × 400
- 500 × 500
- 600 × 600

実験 2

動的位置補正機能を追加することによって，Leaflet を用いた既存システム [8] と比べて，処理速度や応答速度に影響を与えていないか調査する．提案手法において多数のマーカを配置し，Focus+Glue 領域をスクロールして動的位置補正機能を繰り返し作動させたときの，1 秒間の平均フレーム数を測定する．比較対象として，Leaflet のみを用いたマップ（従来手法）を用意し，同様に多数のマーカを配置して画面をスクロールした際の，1 秒間の平均フレーム数を測定する．これにより，提案手法を用いた場合と用いなかった場合の 2 つの場合においてフレームレートの変化を調査する．実験では，マーカの数を 1 から 3,000 まで変化させ，各マーカを特定の領域内にランダムに配置する（**図 11**）．測定にあたっては Stats.js [29] を用いる．

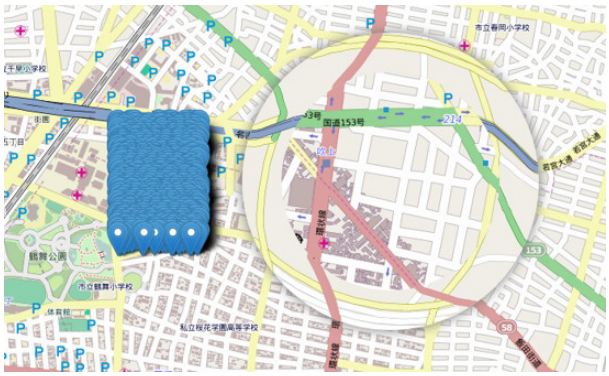


図 11 実験 2 の例
Fig. 11 Example of experiment 2.

表 3 質問内容
Table 3 Questionnaire.

番号	質問内容
1	Focus の描画速度にストレスを感じない
2	注目点に Focus+Glue 領域を簡単に作成できる
3	Focus+Glue 領域を作成・削除が簡単にできる
4	地点検索機能は使いやすい
5	直感的に操作できる
6	サイズ変更がしやすい

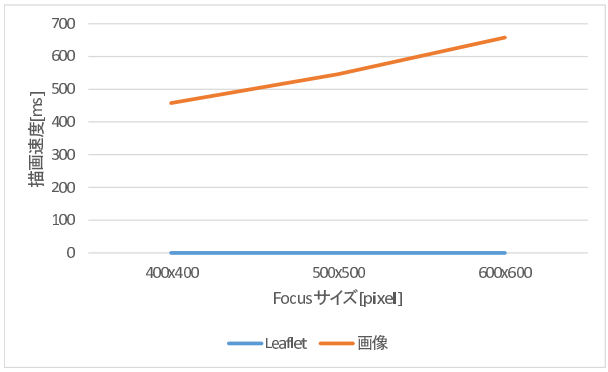


図 12 実験 1 の結果
Fig. 12 Results of experiment 1.

実験 3
本学学生 5 名に iPad で動作する提案システムを自由に使用してもらい，**表 3** の質問に答えてもらう．評価は 5 段階評価で，1 は「そうは思わない」，5 は「そう思う」としている．

6.2 結果

実験 1 結果を**図 12** に示す．実験 1 について，従来手法は，表示する Focus のサイズに比例して 400 × 400 では 458 ms，500 × 500 では 546 ms，600 × 600 では 656 ms と表示するまでの平均時間が増加した．提案手法では，サイズによらず平均で 1 ms を下回る結果となった．

実験 2 結果を**図 13** に示す．提案手法ではマーカが 500 個までは平均フレーム数がおよそ 30 fps 以上であった．

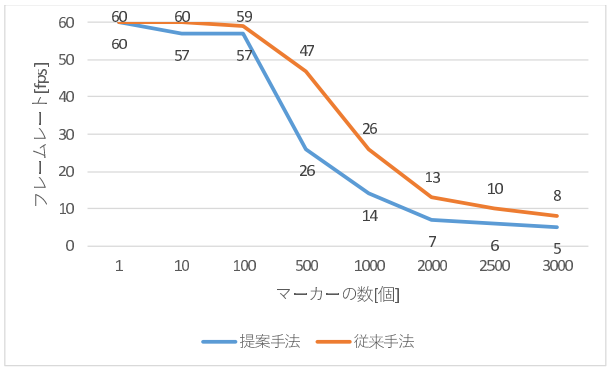


図 13 実験 2 の結果
Fig. 13 Results of experiment 2.

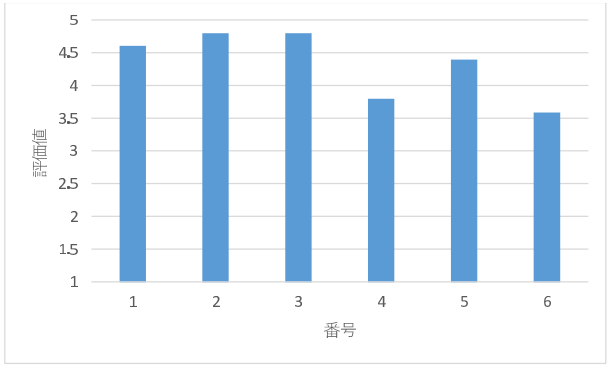


図 14 実験 3 の結果
Fig. 14 Results of experiment 3.

マーカが 500 個に達すると提案手法は従来手法の半分のフレームレートとなり，1,000 個を超えると両者とも fps が 30 を下回る結果となった．

実験 3 結果を**図 14** に示す．1 から 6 の項目すべてで評価点が 3 を超えたが，6 の「サイズ変更がしやすい」では高評価と低評価で分かれる結果となった．

6.3 考察

実験 1 従来手法は表示する場所が決まってからサーバで画像を生成するため，表示までに時間がかかってしまう．それに対し提案手法では，Leaflet のキャッシュ機能により周辺の地図画像をあらかじめ読み込んでおけるため，表示するまでの時間はサイズによらず平均で 1 ms を下回った．ユーザが使ううえでレスポンスは非常に大事な項目であり，表示までのラグがほとんどない提案手法は優れているといえる．なお，既存の魚眼マップに関する研究 [1], [2], [3], [4], [5], [6], [7], [13], [14], [15], [16], [20] を Web マップサービスとして実現するためには，画面全体をサーバ側で動的に描画する必要がある．よって，Focus+Glue 領域のみサーバ側で動的に描画する必要がある従来手法よりも，さらに遅い応答速度となることが想定できる．

実験 2 マーカ数が 100 個までは，提案手法と Leaflet

のみで動作させた従来手法とではほとんど差が生じなかった．提案手法はマーカ 500 個までであればほぼ平均フレーム数 30 fps 以上で動作するため実用的といえる．両者マーカが 1,000 個を超えると平均フレーム数は低くなるが，従来手法に比べ提案手法はフレームレートが約 6 割に下がった．しかし，通常の用途ではほとんど問題にならないと考えている．

実験 3 図 12 を見ると，「Focus の描画速度にストレスを感じない」「注目点に Focus+Glue 領域を簡単に作成できる」「Focus+Glue 領域の作成・削除が簡単にできる」「直感的に操作できる」の 4 項目において評価値はそれぞれ 4.6，4.8，4.8，4.4 であることが分かる．これより提案手法はユーザにとって使いやすく描画速度は十分であるといえる．しかし被験者からは「Focus と Glue のサイズ変更をするときにうまく縁をドラッグできずイライラした」とのコメントもあり，サイズ変更の手法は今後改善が必要である．「地点検索機能は使いやすい」という項目では 3.8 という結果が得られた．提案システムの地点検索機能は地図の中心からキーワードに当てはまるものを近い順に 15 件表示する固定された機能であり，検索オプションなど設定できないといったところが評価結果に表れているのではないかと考える．検索機能とその結果を用いた Focus+Glue 領域の表現方法については今後の課題である．

7. おわりに

本論文では，構造化した 2 つの Leaflet を有し，Focus・Glue・Context の異なる座標系のマルチレイヤ構造を持つ Web マップシステムを提案した．さらに，2 つの Leaflet 間を連携させて整合性を保つ機能，および，地図オブジェクトの描画・移動操作を監視して動的に位置補正する機能を提案し，これにより，可変スケールマップをベースにした位置情報サービスの実現を容易にした．今後は Leaflet のプラグインとして動作可能な BSD ライセンスでの公開を目指している．課題としては Glue の描画速度の向上や 2 つの Focus+Glue 領域が重なったときの各領域の挙動の制御法などがあげられる．

謝辞 本研究は JSPS 科研費 26330136，25700009，および，総務省 SCOPE の助成を受けたものです．この場を借りて，感謝の意を表します．

参考文献

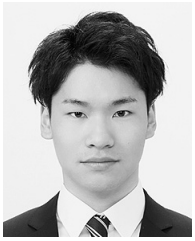
[1] Furnas, G.W.: Generalized Fisheye Views, *Proc. CHI'86*, pp.16–23 (1986).
[2] Harrie, L., Sarjakoski, L.T. and Lehto, L.: A Mapping Function for Variable-Scale Maps in Small-Display Cartography, *JGE*, Vol.4, No.2, pp.111–123 (2002).
[3] Craig, P., Chen, H. and Houssen, F.: A Task Based Evaluation of Fisheye Maps for Mobile Navigation, *1st Conference on Emerging Topics in Interactive Systems* (2016).
[4] Takahashi, N.: An Elastic Map System with Cognitive Map-Based Operations, International Perspectives on Maps and Internet Vol.1, Peterson, M.P. and Liu, J. (Eds.), Lecture Notes in Geoinformation and Cartography, Springer-Verlag (Feb. 2008).
[5] Yamamoto, D., Ozeki, S. and Takahashi, N.: Focus+Glue+Context: An Improved Fisheye Approach for Web Map Services, *Proc. ACM SIGSPATIAL GIS 2009*, pp.101–110 (2009).
[6] 加藤史也，山本大介，高橋直久：任意形状 Focus 生成機能を有する Focus+Glue+Context マップシステムの実現，電子情報通信学会技術研究報告，DE，データ工学，Vol.112, No.346, pp.119–124 (2012).
[7] Mizutani, H., Yamamoto, D. and Takahashi, N.: A Fusion of Multiple Focuses on a Focus+Glue+Context Map, *Proc. International Conference on Intelligent Interactive Multimedia Systems and Services* (*IIMSS 2012*), pp.11–21 (2012).
[8] Yamamoto, D., Murase, M. and Takahashi, N.: Fisheye Map Using Stroke-based Generalization for Web Map Services, *IEICE Trans. Information and System*, Vol.E101-D, No.1, pp.171–180 (2018).
[9] OpenStreetMap Japan, available from ⟨https://openstreetmap.jp/⟩ (accessed 2018-06-28).
[10] Leaflet - a JavaScript library for interactive maps, available from ⟨http://leafletjs.com/⟩ (accessed 2018-06-28).
[11] OpenLayers, available from ⟨https://openlayers.org/⟩ (accessed 2018-06-28).
[12] Gibson, R. and Erle, S.: Chapter 3: Mashing Up Google Maps, Google Maps Hacks, O'Reilly (2006).
[13] Lynch, K.: *The image of the city*, MIT Press (1960).
[14] Gould, P. and White, R.: *Mental Maps*, Penguin Books Ltd., Harmondsworth, Middlesex, England (1974).
[15] Jenny, B. and Hurni, L.: Studying cartographic heritage: Analysis and visualization of geometric distortions, *Computers & Graphics*, Vol.35, pp.402–411 (2011).
[16] Brainerd, J. and Pang, A.: Interactive map projections and distortion, *Computers & Geosciences*, Vol.27, pp.299–314 (2001).
[17] Sarkar, M. and Brown, M.H.: Grapahical Fisheye views of graphs, *Proc. CHI '92*, pp.83–91 (1992).
[18] Sarkar, M., Snibbe, S.S., Tversky, O.J. and Reiss, S.P.: Stretching the Rubber Sheet: A Metaphor for Viewing Large Layouts on Small Screens, *Proc. UIST'93*, pp.81–91 (1993).
[19] Zaslavsky, I.: Online cartography with XML, Maps and the Internet, pp.171–196 (2003).
[20] Wang, Y.S. and Chi, M.T.: Focus+Context Metro Maps, *IEEE Trans. Visualization and Computer Graphics*, Vol.17, No.12, pp.2528–2535 (2011).
[21] Pikorec, M., Sluban, B. and Muc, T.: MultiNets: Web-Based Multilayer Network Visualization, *Proc. ECML PKDD 2015: Machine Learning and Knowledge Discovery in Databases*, pp.298–302 (2015).
[22] 岡 恵，福田 豊：招待講演 航空交通のオープンデータとその活用，電子情報通信学会技術研究報告，Vol.117, No.301, pp.95–100 (2017).
[23] 早川浩平，早川知道，伊藤孝行：OpenStreetMap を活用したイベント情報共有支援システムの試作，情報処理学会研究報告，知能システム（ICS），Vol.2013-ICS-172, No.4, pp.1–6 (2013).
[24] 河村郁江，伊藤孝行：郷土食による地域理解支援シス

テム「もちマップ」の試作，人工知能学会研究会資料，SIG-SWO-041-08, pp.1489–1490 (2018).
[25] 大向一輝：オープンデータ活用：1. オープンデータとLinked Open Data，情報処理，Vol.54, No.12, pp.1204–1210 (2013).
[26] YOLP（地図）- Yahoo!デベロッパーネットワーク，入手先 ⟨http://developer.yahoo.co.jp/webapi/map/⟩（参照2018-02-18）.
[27] jQuery, available from ⟨https://jquery.com/⟩ (accessed 2018-02-18).
[28] Bootstrap, available from ⟨https://getbootstrap.com/⟩ (accessed 2018-02-18).
[29] Stats.js, available from ⟨https://github.com/mrdoob/stats.js/⟩ (accessed 2018-02-18).

推薦文

本論文は，筆者らの先行研究であるWebマップシステムEMMAを拡張して最新のWebマップ技術などによる機能拡張などのカスタマイズを容易にしたOpen EMMAシステムに関する論文である．提案手法は高度交通システムにおいて多くの場合に重要な役割を果たす地図の有用性を高める手法であり，プロトタイプシステムを実装して実証評価を行っている．高度交通システムに関する有用な技術であり，本論文を推薦する．

（高度交通システムとスマートコミュニティ研究会
主査　清原良三）



竹内 健祐

1994年生．2013年愛知県立岡崎北高校卒業．2017年名古屋工業大学工学部情報工学科卒業．同年名古屋工業大学大学院博士前期課程に進学し情報工学専攻に在学．現在，Webマップシステムの研究に従事．

山本 大介　（正会員）

2003年名古屋大学工学部電気電子・情報工学卒業．2008年同大学大学院情報科学研究科博士課程修了．2008年名古屋工業大学工学研究科助教．2011年同准教授．地理情報システム，マルチメディア，Webサービスに関する研究に従事．日本データベース学会，人工知能学会，日本バーチャルリアリティ学会，IEEE各会員．博士（情報科学）．

高橋 直久　（正会員）

1974年電気通信大学応用電子工学科卒業．1976年電気通信大学大学院応用電子工学専攻修了．1976年日本電信電話公社（現NTT）武蔵野電気通信研究所入所．1987年工学博士（東京工業大学）．NTT基礎研究所，ソフトウェア研究所，未来ねっと研究所を経て，2001年名古屋工業大学工学部電気情報工学科教授．2004年同大学大学院工学研究科ながれ領域（情報工学専攻/情報工学科）教授．2017年同大学名誉教授，国際音声言語技術研究所プロジェクト教授．現在，時空間情報処理，音声対話システム，ネットワーク診断システム，eラーニング・システム等の研究に従事．電子情報通信学会，日本データベース学会，IEEE，ACM各会員．本会終身会員．