周回道路データベースの構築とそれに基づく高速な周回道路導出手法

伊藤　広記† 　山本　大介†† 　高橋　直久††

† 名古屋工業大学第一部工学部情報工学科　〒 466–8555 愛知県名古屋市昭和区御器所町
†† 名古屋工業大学大学院工学研究科情報工学専攻　〒 466–8555 愛知県名古屋市昭和区御器所町
E-mail: †hito@moss.elcom.nitech.ac.jp, ††{daisuke.yamamoto,naohisa}@nitech.ac.jp

あらまし　我々は，注目したい拡大領域 (Focus) と周辺領域 (Context) を同時に表示可能な Web マップシステムである Emma を開発してきた．しかしながら，注目領域を適切に内包する Focus を生成するためには，ユーザが複数ステップの操作を経て調整する必要がある．注目領域を取り囲む周回道路を導出し，周回道路に内包する Focus を生成することにより解決できるが，その導出には時間を要する．そこで，本稿は，全ての周回道路を正確に，かつ導出回数を少なくし効率よくデータベースに格納する手法と，それに基づく高速な周回道路導出手法を提案する．

キーワード　Emma，周回道路

Creating a database of roads around the object and method of fast derivation of the roads around the object by using database

Hiroki ITO†, Daisuke YAMAMOTO††, and Naohisa TAKAHASHI††

† Dept. of Computer Science , Faculty of Engineering , Nagoya Institute of Technology
Gokiso,Showa, Nagoya, Aichi, 466–8555 Japan
†† Dept. of Computer Science and Engineering,Graduate School of Engineering,Nagoya Institute of Technology
Gokiso,Showa, Nagoya, Aichi, 466–8555 Japan
E-mail: †hito@moss.elcom.nitech.ac.jp, ††{daisuke.yamamoto,naohisa}@nitech.ac.jp

1. はじめに

我々は，Web マップシステム Emma [1] [2] を開発してきた．Emma とは，図 1 に示すように，注目地域を拡大表示する領域 (Focus)，周辺地域を表示する領域 (Context)，Focus と Context の歪を吸収し両者を結合する領域 (Glue) からなり，全体的な位置関係を把握しながら，注目する領域の詳細情報を一度に確認できる利点がある．

しかしながら，Emma を用いて公園などの注目領域に対して Focus を生成する際，注目領域を内包するように Focus の大きさや位置を調整したい場合，ユーザが Foucs の大きさや中心位置と尺度を調整する作業が必要となる問題があった．

この問題に対し，我々は公園等の地図オブジェクトを取り囲む道路 (以下，周回道路) を獲得し，図 2 に示すようなその周回道路を内包する Focus を生成することにより解決を試みた. [3] [4]

また，周回道路だけでなく，その周辺も Focus に収めたい要求がある．この要求に対しては，図 3 に示すような周回道路をさらに取り囲む周回道路である拡大周回道路を計算し，図 4 に示すような，注目地域だけでなく注目地域の周辺区画をも Focus に納めることにより対応する．繰り返し拡大周回道路を求めることによって，段階的により広い大きさの拡大周回道路を求めることも可能である．さらに，あらかじめ公園や施設などの

図 1　Emma を用いた地図の例

図 2　注目領域の周回道路を内包する Emma

地図オブジェクトの経度緯度座標から周回道路を計算しデータベースに記録しておくことによって，地図オブジェクトの名称から高速に周回道路を求めることも可能になった．

図 3　注目領域の周回道路と対応する拡大周回道路

図 4　注目領域の拡大周回道路を内包する Emma

しかしながら，この従来手法を実際のシステムに適用した際に，以下のような問題が新たに生じた．

問題 1　オブジェクト名称に対応する周回道路がデータベースに登録されていない場合や，任意の緯度経度座標に対する周回道路の導出には，計算時間を要する．

問題 2　とりわけ，より広い範囲の拡大周回道路の導出には，多大な計算時間を要する．

本稿では，これらの問題を解決するために，任意の緯度経度座標に対応した周回道路データベースを構築し，任意の緯度経度座標から高速に周回道路，及び，拡大周回道路を獲得するシステムを提案する．

提案システムは以下の特徴を持つ．

特徴 1　道路データのすべてのリンクに対して時計回りと反時計回りの周回道路をタビュレーション手法により計算する周回道路データベース作成機能を実現する．

これにより，すべての周回道路をもれなく効率的に計算し，データベース化することが可能になる．

特徴 2　あらかじめ，周回道路を内包する最小の矩形領域を周回道路データベースに登録する機能を実現する．また，データベースに登録した矩形領域を用いて指定された地点に対する周回道路を絞り込むことにより効率的に周回道路を探索する機能を実現する．

これにより，指定した地点に対する周回道路を高速に求めることが可能になる．

特徴 3　周回道路データベースに登録されている周回道路を組み合わせて拡大周回道路を導出する機能を実現する．

これにより，指定された点に対する拡大周回道路を高速に求めることが可能になる．

2.　従来システムでの周回道路の導出

まず，道路データの構造について述べる．道路データはリンク，セグメント，ノードという 3 種類のデータからなる．それぞれ，表 1，3，4 のような形式でデータベースに格納されている．

図 5 のように，リンクは，隣接する二つの交差点の Head ノードと Tail ノードを Tail ノードから Head ノードに向かう有向枝であり，セグメントは道路形状に沿った有効枝である．これらは，表 2 の形式のデータによって対応付けられる．また，周回道路や拡大周回道路は表 5 のように，複数のリンクから構成される．

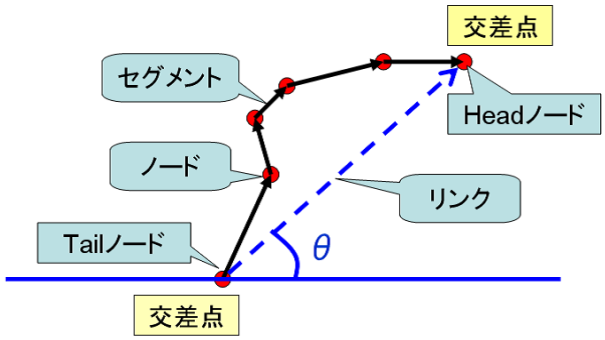


図 5　道路データの構造

2.1　周回道路探索手法

従来手法では，任意の緯度経度座標 P を取り囲む周回道路を以下のステップにて導出した．

具体的には，道路データベースから，緯度経度座標 P を中心とした，十分な大きさを持つ矩形範囲のリンク集合 L を獲得し，そのリンク集合 L と，緯度経度座標 P に対して以下の処理を適用する．

表 1　リンクデータ

項目名	型	説明
リンク ID	int	リンクを一意に識別する ID
角度	double (-180〜180)	西から東を 0 度とした Tail ノードから Head ノードを結ぶ角度
Tail ノード ID	int	リンクの始点となるノードを一意に識別する ID
Head ノード ID	int	リンクの終点となるノードを一意に識別する ID

表 2　リンクとセグメントを一致させるデータ

項目名	型	説明
リンク ID	int	リンクを一意に識別する ID
セグメント ID	int	セグメントを一意に識別する ID

表 3　セグメントデータ

項目名	型	説明
セグメント ID	int	セグメントを一意に識別する ID
角度	double (-180〜180)	西から東を 0 度とした Tail ノードから Head ノードを結ぶ角度
Tail ノード ID	int	セグメントの始点となるノードを一意に識別する ID
Head ノード ID	int	セグメントの終点となるノードを一意に識別する ID

表 4　ノードデータ

項目名	型	説明
ノード ID	int	ノードを一意に識別する ID
緯度	double	ノードの緯度
経度	double	ノードの経度

表 5　周回道路データ

項目名	型	説明
周回道路 ID	int	リンクを一意に識別する ID
リンク ID1，2，⋯,m	text	周回道路を構成するリンク ID をカンマにて区切って格納

STEP1　リンク集合 L に対して，経度緯度座標 P から最も近い位置にあるリンクを求め，初期リンクとする．

STEP2　座標 P とリンクの位置関係から座標 P を囲むような回転方向 (左回り，右回り) を求める．

STEP3　回転方向に応じて，リンクの Head ノードもしくは，Tail ノードと共有するリンクを求める．

STEP4　図 6 のように，左回りの場合にはリンク方向に対して最左枝，右回りの場合には最右枝を選択して深さ優先探索をする．

この時，途中リンクの選択には，途中リンクの方向にかかわらず，初期リンクの方向に沿ってたどっていく．

STEP5　STEP3，4 を繰り返し，初期リンクの Tail ノードと一致したとき，周回道路探索が成功する．

また，全ノード探索しても初期リンクの Tail ノードに到達しない場合や，初期リンクの Head ノードと一致した場合は終了する．

これにより周回道路を求めることが可能になる．ただし，周回道路が大きくなりすぎ，あらかじめ読みだした矩形範囲を超える場合は，さらに大きな矩形範囲を読み出して，STEP1 から処理を適用し直す必要がある．

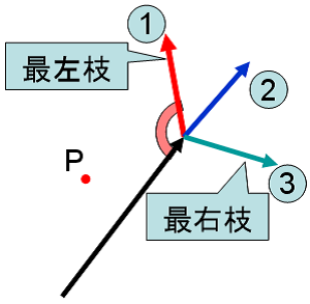


図 6　STEP4 における左回り探索時の選択例

2.2　拡大周回道路探索手法

従来手法にける拡大周回道路探索手法は，元となる周回道路 R に対して，以下のステップを図 2.2 のように適用する．

STEP1　周回道路 R に含まれる全てのリンク L に対して，左回りと右回りの周回道路探索を適用し，その結果得られるリンク集合を L' とする．

STEP2　L' のうち，周回道路 R に含まれるリンクを除去する．

STEP3　L' に含まれるリンク集合のみを用いて，周回道路探索を適用する．その結果得られた周回道路が求めるべき拡大周回道路である．

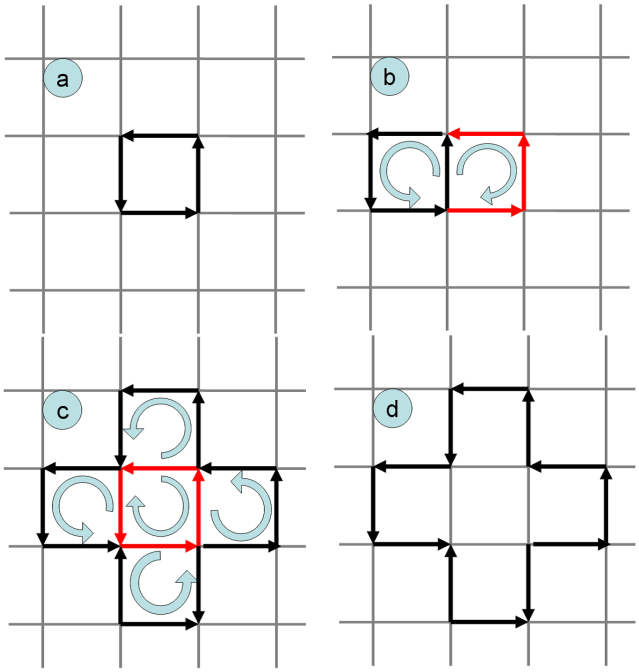


図 7　拡大周回道路探索の実行例

この動作を拡大したいレベル数と同回数繰り返すことにより，任意のレベルの拡大周回道路を求めることが可能になる．

3.　本システムの概要

図 8 に提案システムの構成図を示す．

本提案システムは，周回道路データベースを構築する機能及び周回道路を獲得する機能，拡大周回道路を生成する機能の 3 つの機能を持つ．それぞれの機能については，次章以降説明する．

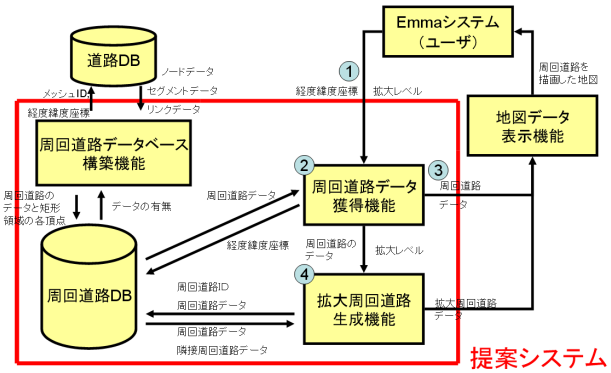


図 8　提案システムの構成図

4.　周回道路データベースの構築法

従来手法では，指定した緯度経度座標からそれを取り囲む周回道路を探索する手法であるため，地図上の全ての周回道路を導出することは効率的に出来なかった．

例えば，地図上を 1m などの細かい格子状に区切り，その全ての交点において従来手法を用いて周回道路を計算すれば，漏れなく周回道路を導出することは可能になるが，非常に時間がかかる．逆に，1km などの荒い格子状に区切れば高速に周回道路を導出することは可能になるが，小さな周回道路などが発見されないジレンマがある．そこで，本章では，周回道路と拡大周回道路を高速に導出するために，道路リンク情報から効率良く周回道路データベースを構築する手法を提案する．

また，図 9 で示すように，周回道路を探索する方向が片方向のみであると，探索されない周回道路が出来る可能性がある．そのため，両方向に周回道路を探索する必要があるが，既に導出された周回道路を再び探索する場合が有り効率が良くない．さらに，複数の道路リンクから成る周回道路の場合，その道路リンク数に応じて何度も周回道路を探索する場合がある．そこで，タビュレーション（索表計算）により，複数回同じ周回道路を探索しないように工夫した．

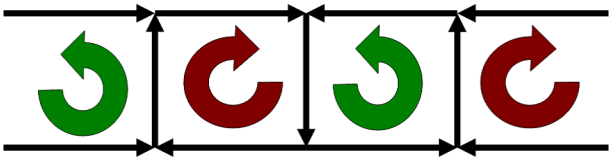


図 9　片方向探索のみでは探索洩れが起きる道路データの例

4.1　周回道路データベースの生成手法

周回道路データベースの生成手法について述べる．

4.1.1　データベースの定義

はじめに，周回道路データベースを構築する際に必要となるテーブルについて説明する．

周回道路を構築するには，以下のテーブルを使用する．

- リンク
- セグメント
- ノード
- 周回道路テーブル
- 隣接周回道路テーブル

表 6　隣接周回道路テーブルのデータ構造

項目名	型	説明
リンク ID	int	リンクを一意に識別する ID
左隣接周回道路 ID	int	リンク方向に対して左に隣接する周回道路 ID
右隣接周回道路 ID	int	リンク方向に対して右に隣接する周回道路 ID

表 7　周回道路テーブルのデータ構造

項目名	型	説明
周回道路 ID	int	周回道路を一意に識別する ID
最東点	double	周回道路を内包する矩形領域の緯度の最高値
最西点	double	周回道路を内包する矩形領域の緯度の最低値
最北点	double	周回道路を内包する矩形領域の経度の最高値
最南点	double	周回道路を内包する矩形領域の経度の最低値
リンク ID1，2，･･･,m	text	周回道路を構成するリンク ID をカンマにて区切って格納

リンク，セグメント，ノードは，2 章に示した通りである．

周回道路テーブルのスキーマは表 7 に示すように，周回道路 ID，矩形領域の各頂点，構成するリンク ID のリストから成る．ただし，周回道路 ID は主キーとしユニークである．構成するリンク ID のリストは，周回道路を探索した順番に格納されており，データベースから獲得する際に再構成しやすくした．矩形領域とは，図 10 のように周回道路を内包する最小の矩形であり，矩形の緯度経度の最大値と最小値からなる計 4 点を周回道路テーブルに格納する．

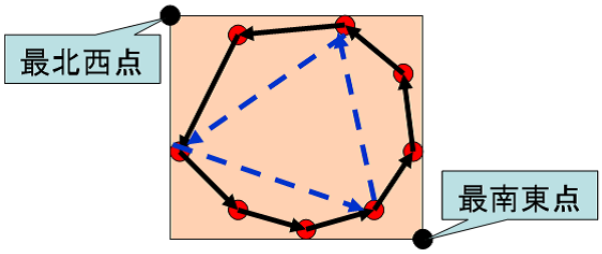


図 10　周回道路を内包する最小の矩形領域

さらに，効率よくデータベースを構築するために，隣接周回道路テーブルを用意する．これは，データベース構築時には，周回道路の重複探索を検出するという効能があり，周回道路の探索時間を減らすという利点がある．具体的には，隣接周回道路テーブルのスキーマは表 6 に示すように，リンク ID，左隣接周回道路 ID，右隣接周回道路 ID の項目から成る．ただし，リンク ID は主キーとし，ユニークである．左隣接周回道路 ID は，数値が入っていた場合，そのリンク ID の左周りに探索し

た周回道路の ID であり，もし数値が入っておらず null となっていた場合は，左回りには周回道路が探索されていないことを表す．右隣接周回道路 ID も数値が入っていた場合と入っていなかった場合は同様の状態を表す．

周回道路テーブル，隣接周回道路テーブルともに，初期状態は空である．

4.1.2 アルゴリズム

全てのリンク L に対して以下の処理を実行する．

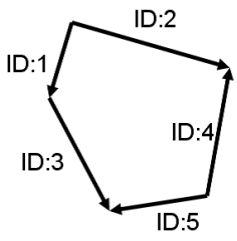
入力 初期リンク L，隣接周回道路テーブル

出力 L を起点とする左右の両方向の周回道路を格納した隣接周回道路テーブル，矩形領域の各頂点と構成するリンク ID のリストを格納した周回道路テーブル

STEP1 隣接周回道路テーブルから，リンク ID の項目が k である行を獲得する．その行の左隣接周回道路 ID が空の場合は，STEP3 へ．ID が存在した場合は，STEP7 へ行き，リンク IDk に対しての左回りの周回道路探索を実行しない．

STEP2 2.1 章で述べた STEP3 以降の手法を用いて，左回りの周回道路探索を実行する．

STEP3 周回道路探索を実行し，探索できなかった場合は，STEP7 へ，探索できた場合，今回探索した周回道路 ID を決め（今回は A とする），探索できた周回道路を構成する全てのリンクのリスト，D に対して，リンクの向きを調べ，隣接周回道路テーブルに格納する．リンクの向きが初期リンク L と同じならば，リンク ID の行の左隣接周回道路 ID に今回探索した周回道路 ID A を書き込む．逆方向であれば，リンク ID の行の右隣接周回道路 ID に今回探索した周回道路 ID A を書き込む．周回道路を探索し，A を 1 として格納した例を図 11 に示す．これを，今回探索した周回道路を構成する全てのリンクに対して実行する．



Link ID	左隣接周回道路ID	右隣接周回道路ID
1	1	Null
2	Null	1
3	1	Null
4	1	Null
5	Null	1

図 11 隣接周回道路テーブルの格納の例

STEP4 隣接周回道路テーブルに書き込んだ後，探索した周回道路を構成するリンクを構成するセグメントを取得する．全てのセグメントのノードの点，経度緯度のそれぞれの最大値，最低値を計算して，矩形領域の四隅の点の座標と，周回道路を構成するリンク ID のリスト D を，今回探索した周回道路 IDA とともに，周回道路テーブルに格納する．

STEP5 右回りに対して，STEP1 から STEP4 までほぼ同様の手順にて実行する．

5. 周回道路導出方式

5.1 周回道路の獲得方式

5.1.1 データベース

今回，周回道路の獲得の際に以下のテーブルを使用する．

- リンク
- セグメント
- ノード
- 周回道路テーブル

5.1.2 アルゴリズム

入力 経度緯度座標 P

出力 周回道路を構成する多角形

STEP1 座標 P を用いて，式 1 で示す SQL 文を用いて，周回道路テーブルにアクセスし，座標 P が矩形の範囲内に存在する周回道路候補集合 As を獲得する．As は，それぞれ構成するリンク ID のリスト LL を持つ．LL は周回道路の探索順になっており，LL のリンクを順に構成していくと，リンクの多角形を構成することができる．

STEP2 LL からリンクを一つずつ取得し，リンクを構成するセグメント ID を順番に表 2 を用いて取得する． 最初に探索したリンクは，常に Tail ノードから Head ノードへの方向に対して探索を行っているので，リンクの Tail ノードから順に，同じ座標のセグメントを探し， 周回道路を構成する多角形ノード集合 PN に対して格納して行き，セグメントをつなげて行く．リンクの Tail ノードに到着したとき，次のリンクを順に探索して格納していく．

STEP3 格納し終えた多角形ノード集合 PN に対して，多角形を構成し，多角形内に座標 P が含まれるかどうか判定する手法を用いる．

STEP4 全ての候補集合 As に対し，STEP3 を実行し，多角形内に P を含む周回道路 PN が，求めるべき周回道路である．

$$select \text{ リンク } ID from \text{ 周回道路テーブル}$$
$$where \text{ 最西点} <= \text{経度 } AND \text{ 最東点} >= \text{経度}$$
$$AND \text{ 最南点} <= \text{緯度 } AND \text{ 最北点} >= \text{緯度;} \quad \cdots\cdots\cdots\cdots\cdots\cdots ①$$

5.2 拡大周回道路の導出方式

5.2.1 データベース

今回，周回道路の獲得の際に以下のテーブルを使用する．

- リンク
- セグメント
- ノード
- 隣接周回道路テーブル
- 周回道路テーブル

5.2.2 アルゴリズム

さらに，任意の緯度経度座標 P に対して，1 レベル上の拡大周回道路を求める手法を提案する．以下の手法をレベル数，繰り返すことで任意のレベル数の拡大周回道路を求めることが可能となる．

入力経度緯度座標 P，拡大レベル N

出力拡大周回道路を構成する多角形

STEP1　5.1 章の手法で P に対して周回道路 A を求め，周回道路 A を構成するリンク ID のリスト LL のリンク ID を全て獲得する．

STEP2　隣接周回道路テーブルに対してリンク ID の項目が k の行の左右の 2 つの隣接周回道路 ID を取得，隣接周回道路 ID リスト RA に格納する．

STEP3　STEP2 を周回道路リンク ID のリスト LL の全てのリンク ID に対し実行し，隣接周回道路 ID リスト RA に順番に格納する．

STEP4　RA から順番に周回道路 ID　A' を取り出し，周回道路リスト AL に A' のリンク ID を格納していく．

STEP5　全ての A' に対し，AL に格納し終えた後，AL から，A に含まれるリンク ID と，重複したリンク ID を取り除く．全て取り除いた AL が，1 レベル上の拡大周回道路のリンク集合となる．

STEP6　得られた AL に対し，5.1 章の手法で多角形を構成する．

STEP1 の入力 A に STEP2 で導出した RA を，STEP1 の入力 LL に STEP5 で用いた AL を入れ，N 回繰り返すことにより，N レベルの拡大周回道路を求めることができる．

6. 評 価 実 験

提案システムの有効性を示すために以下の 2 点について検証を行った

検証 1　データベースに，周回道路の重複がなく，また，探索漏れがないかについての正確性の検証．

周回道路の重複とは，実質的に同一の周回道路を複数回データベースに登録していないかを検証する．

探索漏れは，本来周回道路を生成すべき区画に周回道路がデータベースに登録されているかどうかを検証する．

検証 2　提案システムは，従来手法に比べてどれだけの周回道路探索，拡大周回道路探索の速度が向上したかについての高速性の検証．

6.1 周路道路データベースの正確性

検証手法　地図を描画し、その地図上に周路道路で囲まれた区画をを半透明で塗りつぶすことにより、周回道路が一意に生成されたか，重複がないか，生成できなかった部分がないかを目視で確認した. 重複があれば，より不透明で塗りつぶされる．

検証結果　今回は，2 次メッシュID[523656] と [523657] の 2 つの道路メッシュに対して周回道路データベースを作成した．

結果，[523656] では，1 つの区画が探索されなかった．また，従来の周回道路作成アルゴリズムにおいて，リンクを重複して周回道路を作成してしまう箇所において探索漏れが存在した．また，メッシュの境界をたどるような周回道路を作成できなかった．

しかしながら，表 8 に示したように，[523656] の場合は 99.5%，[523657] の場合は 95.81%にてうまくいく．

ここで書かれている正解 Link 数とは，周回道路を探索できる筈だったリンク数であり，総リンク数ではない．さらに，図 12 のような，道路データベースのミスにより，正しく作成されない場合もあり，これらを手作業で修正するインターフェイスを作成する必要もある．

表 8　作成した周回道路 DB の正誤率

項目	[523656]	[523657]
正解 Link 数	21517	32148
欠落 Link 数	118	1349
正解率	99.452%	95.804%

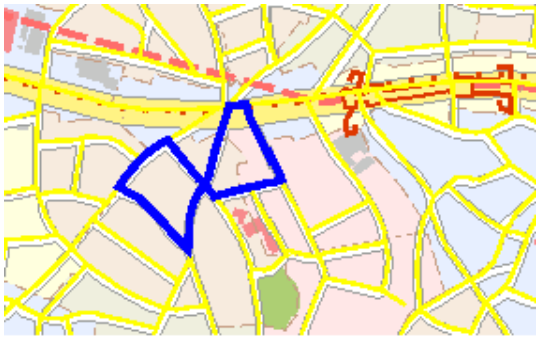


図 12　道路データのミスにより正しく作成できなかった周回道路の例

6.2 既存システムに対しての高速性

検証手法　ランダムに経度緯度座標を与え，周回道路及び拡大周回道路を計算し終えるまでの必要とする時間を比較した．

検証結果　今回は手動で 23 の経度緯度座標を選択して，周回道路を導出し，表示する直前までの時間を測った．結果，表 9 のような結果が得られた．

提案システムでは，平均 78.826ms で既存システムでは 124.913ms であったのに対して，高速化が図られた．また，既存システムの最高所要時間は 562ms であったのに対して，提案システムでは 140ms と 4.0 倍高速であった．特に，リアルタイムシステムでは，最高導出時間の短さがユーザビリティの向上につながると考えられるため，提案手法の利点であると考える．

表 9　周回道路を導出した際の結果

項目	既存システム (ms)	提案システム (ms)
合計導出時間	2873	1813
平均導出時間	124.913	78.826
最低導出時間	15	62
最高導出時間	562	140
分散	17910.17	408.5138

7. おわりに

本研究では，周回道路の導出をデータベースを用いることにより高速に求める手法を提案した．

今後の課題として，拡大周回道路に対して高速化を図ることができなかったため，高速化するための手法を新たに検証する必要がある．また，単一メッシュ内よりデータを取得し，データベースを構築したが，現在メッシュを跨ぐ周回道路を正しく求めることができない，よって，メッシュを跨いでも周回道路を正しく求める手法が必要である．また，現在拡大周回道路を作ると，必ずしも滑らかな形状を取らない．道路の形状によっては，突出してしまう道路が出てきてしまい，必ずしも対象オブジェクトがフォーカス内に適正な大きさで収まらない．よって，より滑らかな形状を取る手法が必要であると考える．

上記の問題を解決する手法の提案，考察，及び評価実験を今後行っていく．

文　献

[1] Naohisa Takahashi: An Elastic Map System with Cognitive Map-based Operations, International Per- spectives on Maps and the Internet, Springer-Verlag,

[2] Daisuke Yamamoto, Shotaro Ozeki, Naohisa Takahashi, Focus+Glue+Context: An Improved Fisheye Approach for Web Map Services, Proceedings of the ACM SIGSPATIAL GIS 2009, Seattle, Washington, pp.101-110, 2009.11 pp.73-87, Feb. 12, 2008.

[3] 福原康平，山本大介，高橋直久：周辺道路の検出による地図オブジェクトのメタデータ生成システム，電子情報通信学会大会，March.20，2009

[4] Daisuke Yamamoto, Kohei Hukuhara, Naohisa Takahashi, A Focus Control Method Based on City Blocks for the Focus+Glue+Context Map, Proceedings of the IEEE 24th Internatinal Conference on Adavanced Information Networking and Applications Workshops (DMWPC 2010), Perth, Australia, pp.956-961,2010.4.21.