フィッシュアイモデルを用いた地図表示システムの提案

篠田 陽平† 井上 潮‡

†東京電機大学工学部 〒101-8457 東京都千代田区神田錦町 2-2
E-mail: †05kc056@ed.cck.dendai.ac.jp, ‡inoue@c.dendai.ac.jp

あらまし 今日，Google マップなどの WebGIS (Web Geographic Information System : Web 地図情報システム) の普及により利用者は容易に地理情報を取得することができるようになった．しかし，これらの WebGIS は従来の紙媒体のそれと比較して視覚的な情報収集方法が向上していない．そこで，コンピュータ上では画像変換が容易にできることに着目し，利用者がいる地点から近い場所を拡大して表示し，遠い場所を収縮させて表示させるフィッシュアイモデルに注目した．本研究ではフィッシュアイモデルを利用した地図情報の表示方法を提案する．

キーワード WebGIS，地理情報，フィッシュアイモデル

1. はじめに

近年，FTTH (Fiber To The Home) に代表される大容量で高速なインターネット環境および高性能で安価な PC が普及してきている．それに伴い，Web2.0 に代表される画像データなどの通信データが大きく，JavaScript，Flash などのクライアントでの処理が大きい Web ページが増加している．WebGIS もその中の一つであり， Google マップをはじめにサイバーマップ・ジャパンの Mapion やインクリメント・ピーの Mapfan など多くの WebGIS サービスが提供されており，利用者は無料で利用することができる．

WebGIS が普及した背景には Google マップのストリートビュー[1]やルート・乗換案内などの紙媒体の地図では実現できず, PC とインターネット環境さえあればだれでも利用できるサービスが出現したということがある．しかし，これらは地図に付加要素を付けているものであり，地図自体からの視覚的な情報収集方法は向上していない．

最近の PC は数値演算が非常に高速であり，画像変換が容易にできる．そこで，地図の表示画面を画像変換することにより視覚的な情報収集能力の向上が可能になる．

本研究では画像の中心付近は拡大表示され，中心から遠ざかるほど収縮されるフィッシュアイモデルを利用し，利用者にとってより多くの情報が必要である中心付近の地理情報を拡大し，逆に有益な情報の少ない周辺の地理情報を収縮させて表示範囲を拡大する新しい地図の表示方法を提案する．

2. 既存研究

2.1. ベースとなる研究

本研究は，我々の研究室で開発中の CMap[2]をもとにしている．CMap は地図情報に Google マップなどが利用している画像データではなく，ベクトルデータである数値地図[3]を利用したクライアントサイド描画手法の WebGIS である．

図 1 に本学，神田キャンパスがある東京都千代田区神田錦町周辺の地図を CMap で表示させた動作画面，および図 2 にそのシステム構成図を示す．

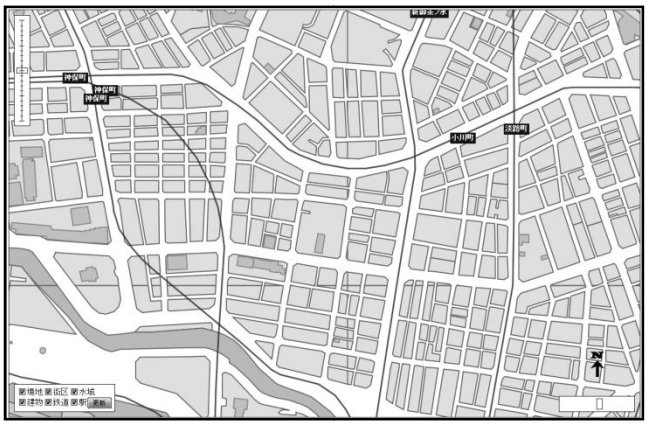

図 1 CMap

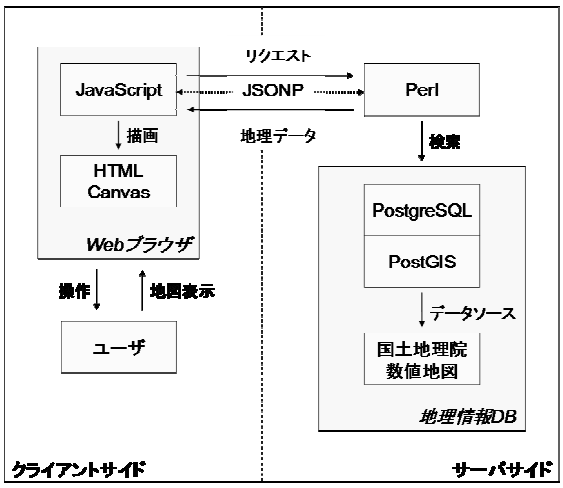

図 2 CMap のシステム構成図

2.1.1. 数値地図

数値地図とは，ベクトルデータとして記録した地図データで，国土地理院が発行している．ベクトル情報であるためコンピュータによる数値変換が容易であるという特徴を持つ．

画像データをコンピュータで画像変換する場合，すべての画素に対して変換する必要があるがベクトルデータの場合，全ての画素にポイントデータは存在しないので変換の処理量が画像データよりも少なくて済む．このベクトルデータの特徴はフィッシュアイモデルの座標変換において有効であるため本研究では数値地図を使用する．

2.1.2. クライアントサイド描画手法

Google マップなどの多くの WebGIS では，あらかじめサーバサイドで地図画像を生成しておき，クライアントからのリクエストがあったときに生成しておいた地図画像を送信するという手法を採用している．それに対し，クライアントからのリクエストがあったときにサーバからは数値データを送信し，クライアントで描画する方法をクライアントサイド描画手法と言う．

両者の描画手法はそれぞれ利点があり，サーバサイドであらかじめ地図画像を生成しておく手法では，新たに地図を描画する必要がないためクライアントサイドのコンピュータ演算能力をあまり必要としないという利点がある．それに対しクライアントサイド描画手法では，数値データを利用しているため数値変換による自由度の高い地図描画が可能となる．これによりサーバサイドであらかじめ地図画像を生成しておく手法では実現が困難である地図の付加機能もクライアントサイド描画手法を用いることにより実現可能になるという利点がある．

具体的な例を挙げると，地図を回転して表示する機能を WebGIS に実装しようとする場合，サーバサイドであらかじめ地図画像を生成しておく手法では，画像データであるため回転させると地名も同時に回転してしまい，これに対応させるには回転する角度ごとに地図画像を用意する必要がある．また，角度がずれる度にサーバから地図画像を取得しなければならない．しかし，クライアントサイド描画手法の場合，地名情報はポイントデータとそれに関連付けられた地名であるため地名の座標が回転しても文字列が傾くことは無い．また，角度がずれる度にサーバからデータを取得する量もデータが存在しない領域だけで良く，通信コストが少なくて済む．

別の例を挙げると，「街区」「道路」「線路」などを切り換えて表示させたい場合，サーバサイドであらかじめ地図画像を生成しておく手法では切り替える度にサーバにリクエストを送らなければならない上にサーバサイドでは多量の地図データを用意しておかなければならない．しかし，クライアントサイド描画手法では差分データを送信するだけで済む．

このようにクライアントサイド描画手法を用いることによって得られる利点は多い．本研究ではフィッシュアイモデルを用いて地図を歪ませて表示させるため，サーバサイドであらかじめ地図画像を生成しておく手法を利用すると多量の地図画像が必要となり現実的では無い．従って，本研究ではクライアントサイド描画手法を利用する．

2.2. 関連研究

フィッシュアイモデルを利用した既存の WebGIS として ALPSLAB の Fish-Eye powered by Emma[4]がある (図 3).

Fish-Eye powered by Emma では利用者が注視している視点に円形の領域を作り，そこに拡大した地理情報を表示させ，他の領域との境界線をフィッシュアイモデルで歪ませて表示させている．しかし，Fish-Eye powered by Emma ではフィッシュアイモデルで歪ませている境界で道路の繋ぎ目がずれており，地名や駅名などの文字情報が切れている．また，歪ませた領域も文字情報が表示されなく，幹線道路などの大きな道路は表示されているが，生活道路などの小さな道路は表示されないなど利便性に欠けている．

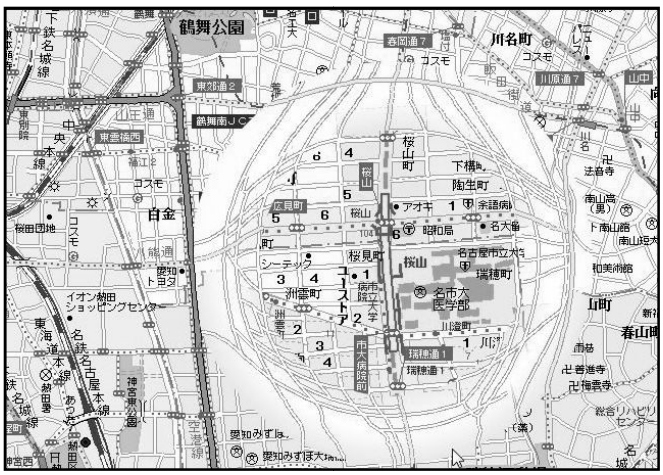

図 3　Fish-Eye powered by Emma

3. 研究目標

本研究では同じ表示領域でより多くの地理情報を表示させることを目標とする．

しかし，それを満たすためにはより多くの地理データを必要とし，それに伴い描画する負荷や通信コストが増加するという欠点がある．そこで，同じ表示領域

でより多くの地理情報を表示させつつ，通信コストを抑え，描画に時間をかけすぎないシステムを設計，構築することが目標である．

4. 目標達成方法

本研究では同じ表示領域でより多くの地理情報を表示させる手段としてフィッシュアイモデルの利用を提案する．フィッシュアイモデルは主にカメラのレンズで使用され，180 度近くの広い視野角を持つことが大きな特徴である．

一般的な地図の場合，地理情報を平面として捉えて平面に表示するため，利用者が注目しているある地点から遠くの別の地点の地理情報を取得したい場合は，画面を移動させるか，画面の表示領域を広げなければならない．しかし，フィッシュアイモデルの広い視野角を利用することによって表示領域を広げることなく複数の地点の地理情報を取得することができる．

5. フィッシュアイモデルのアルゴリズム

本研究で使用したフィッシュアイモデルの画像変換式を以下に記す．

$$x' = \frac{r\,x}{\sqrt{d^2 + x^2 + y^2}}$$

$$y' = \frac{r\,y}{\sqrt{d^2 + x^2 + y^2}}$$

これらの式において，画像の各画素を直交座標で表したとき，x，y は変換前の x 座標，y 座標．また，x’，y’は変換後の x 座標，y 座標の値を表す．r はカメラのレンズから視点までの距離，d は歪みの強さを表す．

しかし，これは画像データにおいて中心を原点とした一般的な直交座標で変換するための変換式である．本研究で扱う HTML の Canvas 要素は左上角を原点とした直交座標系であり，また，サーバから送信される数値データは表示領域の中心からの距離ではなく左上端部からの距離であるため変換式を最適化した．その変換式を以下に記す．

$$x' = \frac{r\left(x - \frac{w}{2}\right)}{\sqrt{d^2 + \left(x - \frac{w}{2}\right)^2 + \left(y - \frac{h}{2}\right)^2}} + \frac{w}{2}$$

$$y' = \frac{r\left(y - \frac{h}{2}\right)}{\sqrt{d^2 + \left(x - \frac{w}{2}\right)^2 + \left(y - \frac{h}{2}\right)^2}} + \frac{h}{2}$$

これらの式において，w と h はそれぞれ表示領域の幅と高さを表す．

6. 実装

6.1. クライアントサイドの実装

CMap の実装を基本としつつ，地図データをフィッシュアイモデルで変換する処理を追加した．図 4 にクライアントサイドでの処理の流れ，図 5 に JavaScript での処理の流れを示す．

サーバから送られてきた JSON 形式の地理データを JavaScript が取得し，「街区」「道路」「線路」などのオブジェクトごとに一つずつ取り出してフィッシュアイモデルを利用して座標変換，そして HTML の Canvas 要素へ描画している．また，CMap ではサーバへのリクエストパラメータに表示領域の高さと幅を指定しているが，フィッシュアイモデルで直交座標を変換するためには表示領域の半径が必要であるのでリクエストパラメータの変更した．

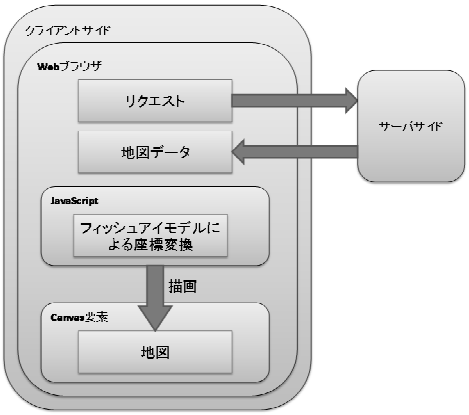

図 4　クライアントサイドでの処理の流れ

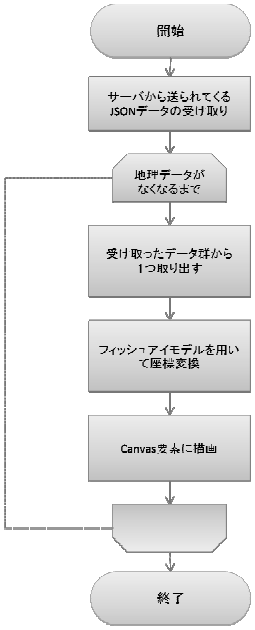

図 5　JavaScript での処理の流れ

6.2. サーバサイドの実装

CMap のプログラムを参考にし，クライアントからのリクエストのパラメータを変更に対応するようにプログラムを修正した．

サーバサイドの環境を表 1 に記し，サーバサイドの処理の流れを図 6 に示す．

クライアントからリクエストが来たら Perl でリクエストパラメータを基に SQL 文を作成し，データベースへ問い合わせをする．そして，データベースから取得した地図データを後述する JSONP 形式に変換してクライアントに送信している．

表 1 サーバ環境

サーバ OS	Ubuntu 8.10 Desktop Edition
Web サーバ	Apache 2.2.8
DBMS	PostgreSQL 8.3
DB 拡張	PostGIS 1.3.3
プログラミング言語	Perl 5.10

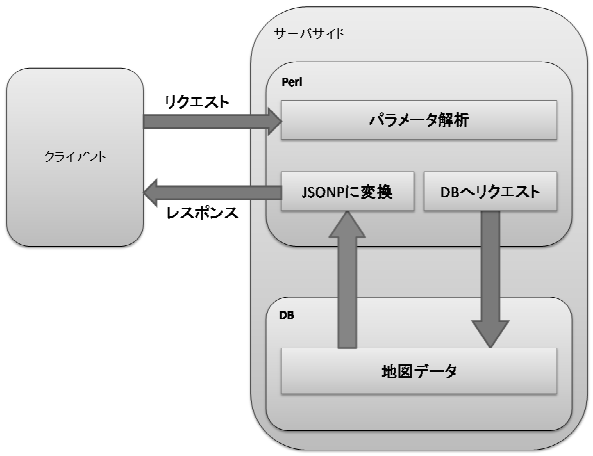

図 6　サーバサイドでの処理の流れ

6.3. 通信方式

通信方式は CMap のそれをそのまま流用した．JSONP による非同期通信で，通信コストを低減するために送信データを gzip 圧縮している．同期通信ではリクエストが送られる度に画面が更新されるため画面の変移を滑らかにすることが困難である．そこで，非同期通信を用いることによりこの問題を解決している．

JSONP（JSON with Padding）とは JSON による軽量なデータ通信方式である．また，JSON（JavaScript Object Notation）とは構造化されたデータを記述するためにテキストによるデータ記述法の一つであり，JavaScript でのオブジェクトを記述する構文と同じであるため人間が理解しやすく，プログラムによる処理が容易であるという特徴がある．また，gzip 圧縮することにより圧縮と解凍に負荷が掛かるというデメリットあるがデータ転送量が約 3 分の 1 になるというメリットを得られる．

実際に送信されるデータの例を図 7 に記す．

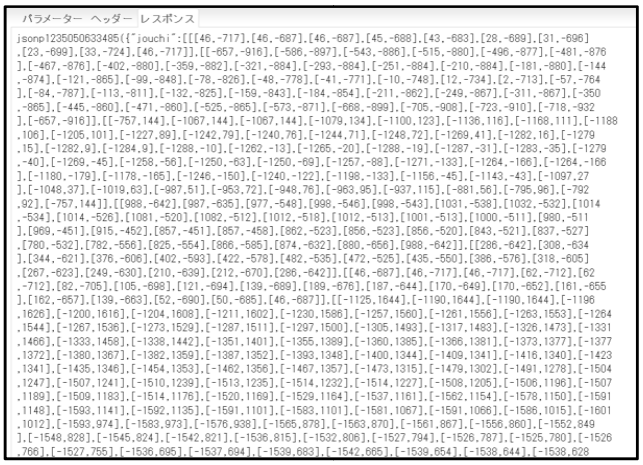

図 7　JSONP レスポンスの例

6.4. プロトタイプ

作成したプロトタイプは CMap の標準的な機能である地図尺度の拡大と縮小，地図の回転をそのまま実装している．画面左上部にあるつまみで地図尺度の拡大と縮小，画面下部のつまみで地図の回転ができる．

また，地図をドラックすることにより地球儀を回すような感覚でスムーズに視点を移動させることができる．

本学，神田キャンパス周辺の地図をプロトタイプで表示させた画面を図 8 に示す．

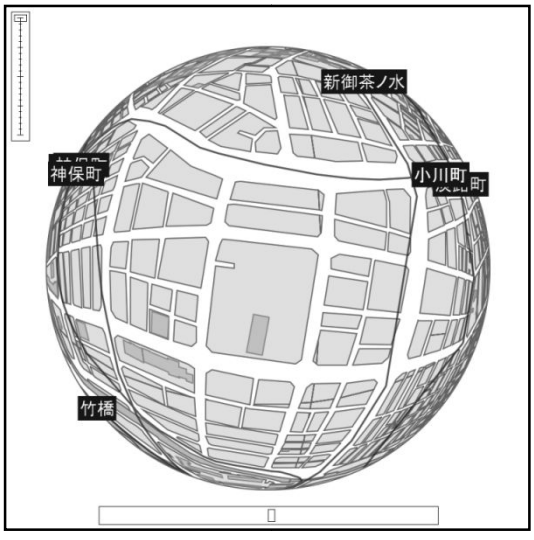

図 8　プロトタイプの表示画面

7. 評価

Web ブラウザにおいて同じ表示領域でどれだけ多くの地理情報を表示させられるか．及び，クライアントでリクエストを送ってから地図描画されるまでどれだか時間が掛かるかを評価した．

7.1. 表示領域

プロトタイプとフィッシュアイモデルを適用していない一般的な地図で表示領域がどう変わるかを二つの地図を表示させ，地図の中心の尺度を同一にした場合と地図の表示領域を同一にした場合の 2 パターンで評価する．その結果を図 9，図 10 に示す．

地図の中心の尺度を同一にした場合，プロトタイプは高さと幅がともに約 330px であるのに対し，フィッシュアイモデルを適用していない地図では 1000px であった．従って，プロトタイプの表示領域はフィッシュアイモデルを適用することにより表示領域を約 9 分の 1 にできることが分かった．

また，地図の表示領域を同一にした場合，利用者が注目する表示領域の中央を拡大表示し，中心から遠ざかるほど収縮されて表示されていることがわかる．

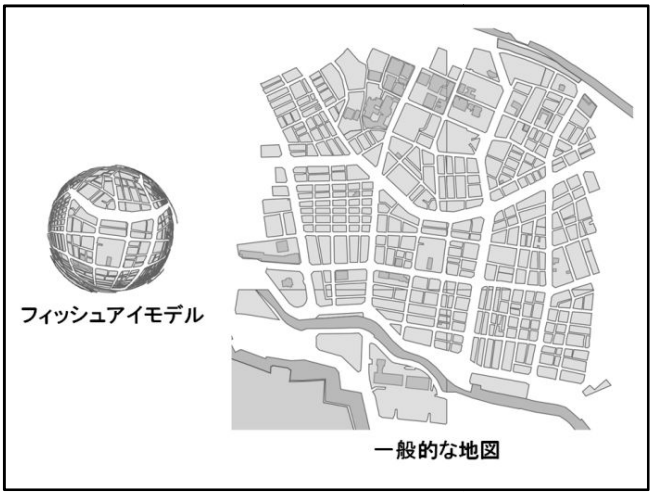

図 9　中心の尺度を同一にした比較

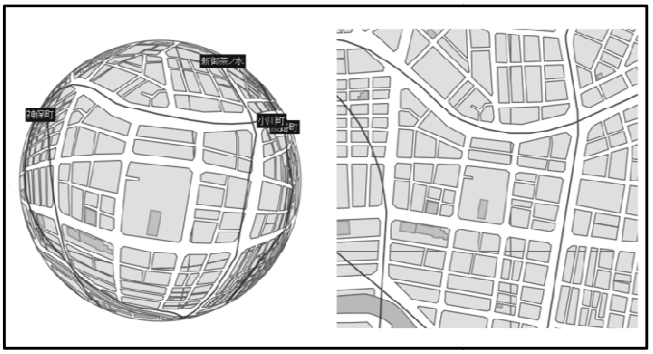

図 10　表示領域を同一にした比較

7.2. 速度測定

表 2，表 3 に示す測定環境においてクライアントに表 4 の Web ブラウザを使用し，地図表示の領域を 800×800 px にしてプロトタイプと CMap で通信に要した時間と地図の描画時間，及びその合計を各 10 回測定し平均値を比較した．結果を表 5 から表 8 に記す．

結果から，どの測定環境においても Google Chrome が高速でありことがわかる．また，測定環境によって通信に要した時間が大きく変わることはないが CPU の周波数が高いほど全体の処理時間が短いことから CPU の性能に大きく依存することがわかる．

処理の内訳を考察すると，環境によって変化するが描画時間は全体の処理の約 3 分の 1 未満であり，通信に要した時間が全体の処理の 3 分の 2 以上を占める．全体に対する描画処理の割合はそれほど大きくなく，また，クライアントサイドの CPU の性能は加速的に向上していくため，パフォーマンスの向上には通信のデータサイズを削減することが重要である．

表 2　測定環境 A

OS	Microsoft Windows Vista Business
CPU	Intel Core2 Duo E8400 @3.00GHz
メモリ	4GB

表 3　測定環境 B

OS	Microsoft Windows Vista
CPU	Intel Core2 Duo SU9300 @1.20GHz
メモリ	4GB

表 4　測定に用いた Web ブラウザ

Web ブラウザ	バージョン
Mozilla Firefox	3.0.6
Google Chrome	1.0.154.48
Apple Safari	3.2.2

表 5　測定環境 A によるプロトタイプ測定

10 回の平均[ms]	Firefox	Chrome	Safari
通信に要した時間	743.9	685.5	698.6
地図の描画時間	134.9	95.4	135.6
合計	878.8	780.9	834.2

表 6　測定環境 B によるプロトタイプ測定

10 回の平均[ms]	Firefox	Chrome	Safari
通信に要した時間	695.5	702.1	722.7
地図の描画時間	264.3	159.1	274.9
合計	959.8	861.2	997.6

表 7　測定環境 A による CMap 測定

10 回の平均[ms]	Firefox	Chrome	Safari
通信に要した時間	568.7	530.5	536.1
地図の描画時間	73.0	45.7	45.3
合計	641.7	576.2	581.4

表 8　測定環境 B による CMap 測定

10 回の平均[ms]	Firefox	Chrome	Safari
通信に要した時間	549.1	542.4	544.3
地図の描画時間	115.3	78.1	71.5
合計	664.4	620.5	615.8

表 9　全体の処理に対する描画時間の割合

描画時間の割合[%]	Firefox	Chrome	Safari
測定環境 A プロトタイプ	15.35	12.22	16.26
測定環境 B プロトタイプ	27.54	18.47	27.56

8. おわりに

8.1. まとめ

本研究ではフィッシュアイモデルを利用することにより，中央付近の地図を拡大して表示し，中央から離れた周辺の地図を収縮させて表示させる新しい地図情報の表示方法を提案および実装をした．その結果，表示領域を一般的な地図の表示方法に比べ約 9 分の 1 にすることができ，表示領域が同一のままより多くの地理情報を表示させることができた．

また，データ通信量と地図描画の速度を測定し評価を行った．その結果，地図の描画は全体の処理の 3 分の 1 未満であり，サーバからデータを取得する処理が全体の処理の 3 分の 2 以上を占めていることが分かりパフォーマンスに多くの影響を与えていることがわかった．しかし，サーバへのリクエストから 1 秒未満で地図の描画ができ十分実用的な速度であることを確認した．

8.2. 今後の課題

前述したように処理の多くの時間を通信に割り当てえいる．従って，どのようにして通信コストを削減するかが今後の課題である．

プロトタイプにはキャッシュの利用を実装していない．従って，キャッシュを利用することによってどれだけ通信コストの削減になるかを今後検証する．

また，フィッシュアイモデルでは地図の尺度を広範囲にすると動作が遅くなるという欠点がある．この問題を解決するためにフィッシュアイモデルの変換式の近似し，処理量を削減するなどシステムの向上をさせる予定である．

文　献

[1] Google マップ ストリートビュー，http://maps.google.co.jp/
[2] 矢島健太郎，山崎優，井上潮，”クライアントサイド描画手法を用いた WebGIS の提案と実装”，DEIM2009 発表予定．
[3] 国土地理院 数値地図 2500.
[4] ALPSLAB，Fish-Eye powered by Emma，http://joint.alpslab.jp/fisheye/