効率的な地図ナビゲーションのための階層的ルートマップの生成

王 方舟　　Yang Li　　五十嵐 健夫*

概要.　ルートマップにおいてルーティング情報を効果的にユーザに提示するためには，曲がり角の詳細な情報などを示すための大縮尺図や，ルート全体の概観示すための小縮尺図など様々な縮尺 (スケール) と表示範囲を持った地図のビューにユーザが効率的にアクセスできる必要がある．一方，一般的なルートマップのブラウジングや印刷システムにおいては，この点が十分に達成されているとは言えない．この問題を解決するため，我々は「ルートツリー」と呼ばれるルートマップの階層構造と，その構造を自動生成するアルゴリズムを提案する．ルートツリーは様々なアプリケーションに応用可能な構造であり，本稿では，インタラクティブなルートマップの閲覧システムである RouteZoom と，ルート情報の印刷のための TreePrint の 2 つのアプリケーションを提案する．我々は RouteZoom に関してユーザスタディを実施し，その可用性を検証した．

1　はじめに

ルートマップとは，2 つの地点間のルートを示し，目的地に到着する為に必要な情報を提供する地図のことである．一般的に，ユーザはルートマップから情報を把握する際に，様々なスケールや表示範囲をもつ複数のビューへのアクセスを必要とする．例えば，ルートの全体を把握するためには小さなスケールのビュー，ルート中の曲がり角や分岐点など POI(Point of Interest) と呼ばれる場所の周辺の詳細な情報を知りたいと思う時は大きいスケールのビューといった具合である．一方，一般的な紙印刷の地図帳や，Google Maps などのシステムに採用されている標準的な地図閲覧手法においては，ユーザは欲しいビューを得るために手動で地図のスケールや閲覧範囲を操作する必要がある．ユーザにとって意味のあるビューは，ルートマップ中の現在地やユーザ自身の興味によって刻一刻と変化するため，このような手動での操作は時として非常に面倒に感じられる．逆に言えば，もしユーザにとって意味のある様々なビューに素早く簡単にアクセスする手法が存在すれば，ルートマップのユーザビリティは大きく向上するであろうと考えられる．

本稿では，上記のようなユーザにとって意味のあるビューへの素早く簡単なアクセスを可能にするために，そのようなビューを事前に抽出し,「ルートツリー」と呼ばれる階層構造として保存する手法を提案する．ルートツリーの例を図 1 に示す．この構造の重要な特徴は主に以下の 3 つである．

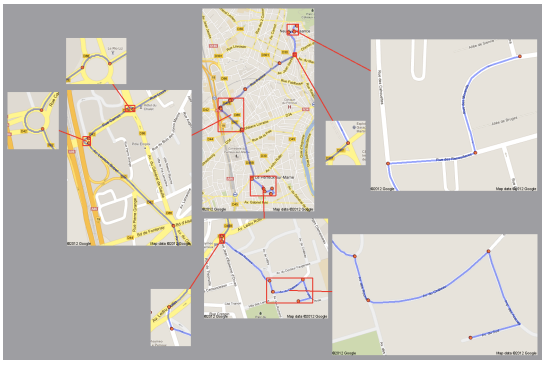

図 1. ルートツリーの例

1. 根ノードはルート全体のビューを表しており，ツリーの各ノードは，親ノードのビューの一部をより大きな縮尺で表すビューになっている．
2. ルート中の任意の部分区間について，必ずその区間を可視化に十分な大きさのスケールで描いているノードが存在する．
3. ノード数および各ノードの表すビューのサイズが適切である．

本稿ではさらに，ルートツリーを自動で計算しその構造を最適化するアルゴリズムを提案する．最適化の際にアルゴリズムに与える評価関数を調整することで，ツリー生成の際に様々な制約を加えることができる．このため，ルートツリーは目的に応じて様々なアプリケーションに応用することができる．その可用性および汎用性を示すため，本稿では更にルートツリーを用いた 2 種類のアプリケーションを提案する．1 つ目の「RouteZoom」はインタラクティブなルートマップの閲覧のためのアプリケーションであり，2 つ目の「TreePrint」はルート情報の印刷のためのアプリケーションである．

* Houshu Oh and Takeo Igarashi, 東京大学大学院 情報理工学系研究科 コンピュータ科学専攻, Yang Li, Google Research

2 関連研究

Agrawala ら [4] は，手書きのルートマップを模倣した抽象化されたルートマップを自動生成するシステムを提案した．生成された地図においては，複数のスケールの情報が一つのビューに融合されている．一方で，ルートの幾何的な形状や道の視覚的な長さといった情報は歪められ，またルート周辺の情報は大幅に削られている．このため，ユーザが道を間違えた際に元のルートに戻ることが著しく難しくなっており，道の形や周囲の建造物などから正確な地図中の位置を特定することも困難になっている．さらに，道の幾何的な形状が歪められているため，いくつかのマップサービスで提供されている航空写真などの機能との相性も悪い．

Karnick ら [11] は，ルートの抽象化を行わずに，ルート全体のビューの周りに一つ一つの POI に対応する詳細なビューを適切な順番で並べることで，複数のスケールの情報を視覚化する手法を提案している．しかし，この手法には，各ビューのスケールが内容によらず固定であるため必要な情報が得られない場合があること，POI の数だけビューが生成されるために似たようなビューが生成され冗長であること，POI がルート全体の一部に密集している場合 POI 同士の距離感や位置関係が把握できなくなる，といった問題がある．また，幾つかの既存のマップサービス [1][2] が提供する類似の印刷機能も，Karnick らの提案手法と同様の問題を抱えている．

複数のスケールをまたいだ地図の閲覧手法に関しても，様々な研究がある．その多くは，Furnas ら [7] の魚眼レンズに代表される Focus + Context の手法を応用したもの [5][6] や，複数のスケールの画面を同時にユーザに見せる Overview + Detail の手法を応用した [8][10] ものである．その他のアプローチとしては，従来の拡大縮小や平行移動のインタフェースを拡張したものがある．五十嵐ら [9] は，画面のスクロールの速さに合わせて自動で地図のスケールを変化させる手法を提案した．Malacria[12] らは，画面上に円を描くというなめらかな操作で地図の拡大縮小およびドラッグを実現するインターフェースを提案した．Robbins ら [13] は閲覧画面を幾つかのエリアに分割し，それらエリアを繰り返し選択することによりズームイン操作を行うことを可能にする ZoneZoom と呼ばれる手法を提案した．一方で，これらの閲覧手法はユーザによる入力の情報のみを利用しており，閲覧する対象である地図のセマンティクスは無視されている．我々の提案した RouteZoom アプリケーションは，ユーザインタフェースは ZoneZoom の提案手法を基にしているが，一方で上記のズームすべき範囲をルート情報から自動抽出しているという点で従来手法と大きく異なる．

3 ルートツリーの生成アルゴリズム

提案するアルゴリズムは，オンラインのルート検索サービス [3] などから得たルート情報を入力とし，ルートツリーを出力とするものである．アルゴリズムは大きく分けて，スケール評価，初期ツリー生成，最適化の 3 つのステップをたどる (図 2)．以下の各節において，それぞれステップの詳細を述べる．

3.1 スケール評価

入力とするルート情報は，ルートの幾何的な形状や各部分の道の種類，POI の位置やその種類などの情報を含んでいる．ルートの形状は，緯度経度座標上のポリラインとして表現され，POI はポリライン上の点として表される．アルゴリズムは，まずこのポリラインを短い線分の断片に分割した上で，スケール値と呼ばれる値を各断片に付与する．この断片をルートセグメントと呼ぶ．スケール値は，各ルートセグメントを地図上で可視化するために必要最小限の地図のスケールを表す値である．提案アルゴリズムでは，1) セグメントを含む道の種類 (高速道路，国道，住宅地の道など)，2) セグメントが POI を含んでいるか否か (含んでいる場合，曲がり角，ジャンクションなどその種類も)，3) セグメントに最も近い 2 つの POI 同士の距離，の 3 つの要素を基にスケール値を機械的に評価する．例えば，高速道路の一部を表すルートセグメントは比較的小さなスケール値が付与され，細い道であったり，曲がり角など POI 周辺を表すルートセグメントはより大きなスケール値を付与される．スケール値は 0 から 20 前後 (地図データにより異なる) の整数で表されビューを生成する際の地図画像の縮尺を決定する．

3.2 初期ツリー生成

アルゴリズムは第 2 ステップとして，初期ツリーと呼ばれるルートツリーの一種を生成する．図 3 に

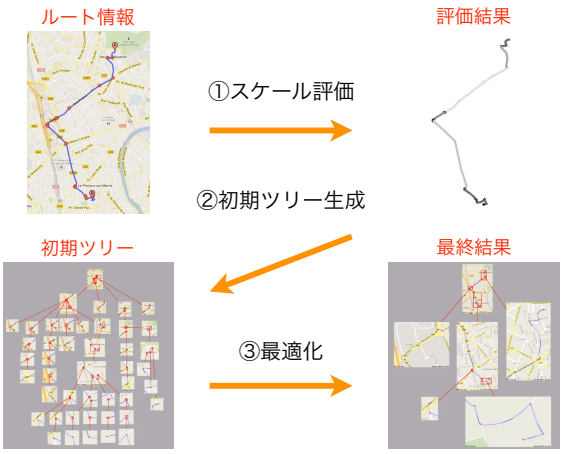

図 2. ルートツリー構築アルゴリズムの概要

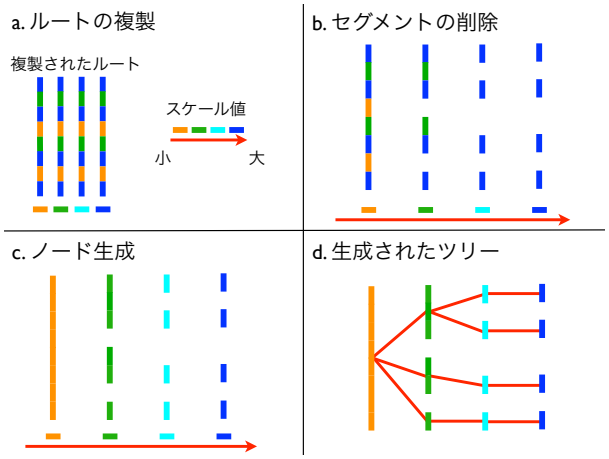

図 3. 初期ツリー構築

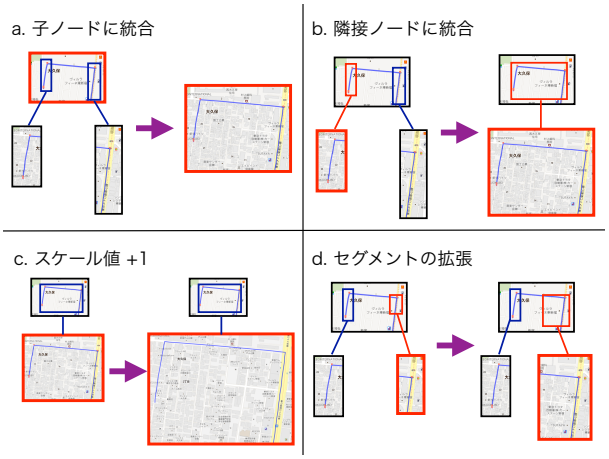

図 4. 最適化時に適応する操作

生成手順の概要を示す．アルゴリズムはまず，それぞれのスケール値に対してセグメント化されたルートを複製する (図 3-a)．次に，複製したそれぞれのルートに対して，対応するスケール値より小さなスケール値を持つルートセグメントを削除する操作を行う (図 3-b)．そして，連続したルートセグメント同士を接続し，対応するスケール値を付与してノードを作成する (図 3-c)．最後に，各ノードをルートセグメントの集合とみなし，その包含関係に基づいて枝で接続することで初期ツリーを生成する (図 3-d)．

3.3 最適化

前ステップにおいて生成された初期ツリーは，ノード数が過剰で，また各ノードの表すビューが非常に小さく，実用性に乏しい．最終ステップとして，アルゴリズムは以下の 4 つの操作を初期ツリーに繰り返し適用することで，ツリーのノード数と各ノードの表す地図画像のサイズを最適化していく．

1. 対象ノードを子ノードに統合する操作 (図 4-a)
2. 対象ノードを隣接するノードに統合する操作 (図 4-b)
3. 対象ノードのスケール値を 1 増やす操作 (図 4-c)
4. 対象ノードの含むルートセグメントを前後に 5 つ拡張する操作 (図 4-d)

ルートツリーの最適化は，遺伝的アルゴリズムにより「操作配列」と呼ばれる配列を最適化していくことで行われる．操作配列は，初期ツリーのノード数の定数倍 (現実装では 7 倍) の長さを持つ整数配列である．配列の各セルは初期ツリーのノードの一つと対応し，その値は対応するノードに対して前述の 4 つの操作のいずれを適応するかを表している．この配列を先頭から走査し，初期ツリーの各ノードに対して操作を適応していくことにより，適切なノード数とビューの大きさをもつルートツリーを初期ツリーから生成することができる．同一の初期ツリーのもとでは，各操作配列は最終結果のルートツリーと一対一に対応していると言える．各操作配列のコストは，生成されたツリーを評価関数で評価した結果とする．ここで用いる評価関数はアプリケーション毎に異なるため，その設計は 5.3 節で改めて述べる．遺伝的アルゴリズムは，1 世代の個体数を S 個として，操作配列を第 I 世代まで最適化することによって最終結果となるツリーを生成する．現実装では，$S = 500, I = 1000$ としている．

4 ルートツリーの生成結果

図 5 に，前節で説明したアルゴリズムによって自動生成されたルートツリーの例を示す．この例においてはロンドンの郊外に位置する West Lodge Park Hotel から Hyde Park までの約 20km のルートを階層化した結果を示している．図 5-a に入力となるルートおよび生成結果の概要，図 5-b および図 5-c に生成結果の一部の拡大図を示す．生成において用いた評価関数は，後述の RouteZoom アプリケーションで用したものに準ずる．結果においては，ルート全体が木構造として適切に構築され，各々のノードに可視化のために適切なスケール値が割り当てられていることが確認できる．

5 アプリケーション

本稿では，ルートツリーの応用例として，インタラクティブなルートマップ閲覧システムである RouteZoom と，ルートマップの印刷のためのシステムである TreePrint の 2 つのアプリケーションを提案する．以下，それぞれのアプリケーションの詳細と，機能を実現するためにルートツリーを生成する際に用いた評価関数の設計について述べる．

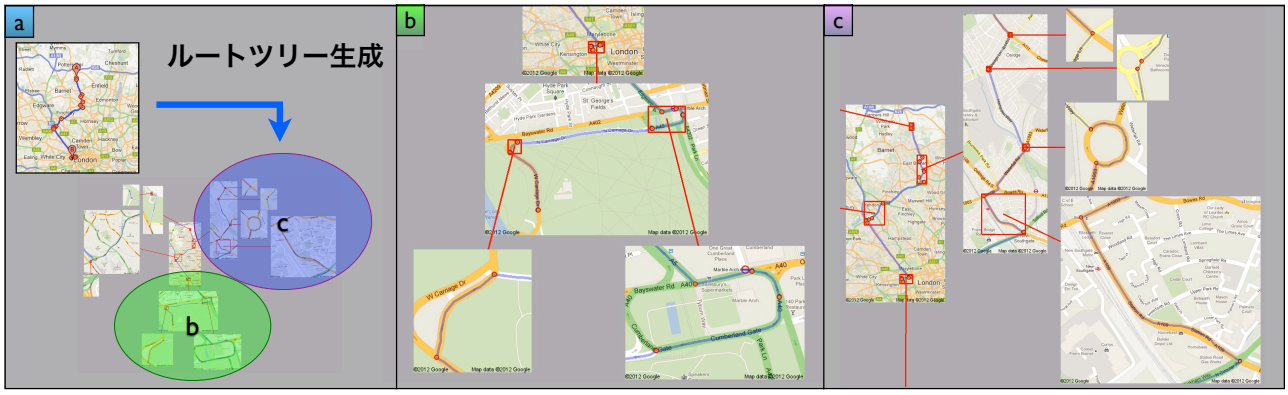

図 5. ルートツリーの生成結果

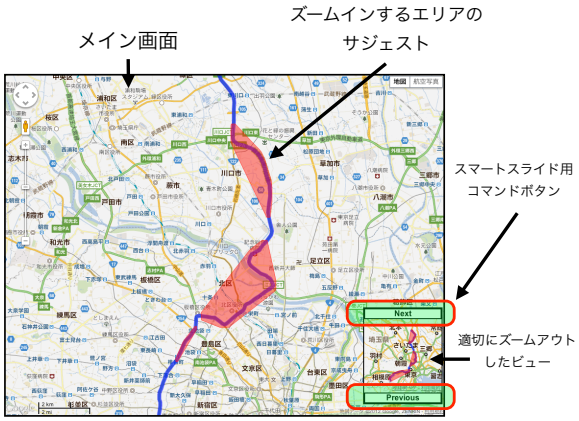

図 6. RouteZoom アプリケーションの UI

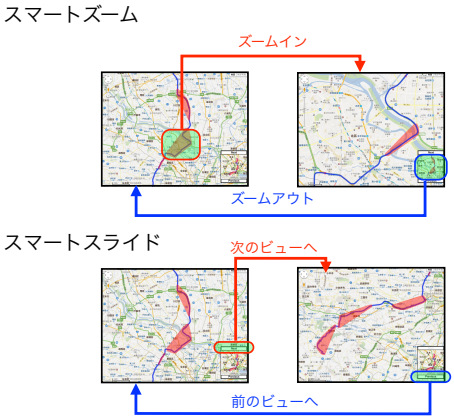

図 7. スマートズームおよびスマートスライドの動作

5.1 RouteZoom

RouteZoom アプリケーションの基本的なインタフェースを図 6 に示す．RouteZoom には 2 つの主要な機能がある (図 7)．

1 つ目のは「スマートズーム」と呼ばれる機能である．RouteZoom では，システムは自動でユーザがズームインしたいであろうと思う範囲をメイン画面に，適切にズームアウトされたビューを右下のサブ画面にサジェストする．ユーザはサジェストされた範囲をクリックないしタップするだけで，そのエリアの詳細なビューを得ることができる．注意すべきは，標準的な地図アプリケーションにおいては，ユーザはズームインした後に手動で表示位置を調整する必要があるのに対して，この機能は地図のスケールだけでなく，その位置をも適宜調整することで適切なビューをユーザに提供することができるという点である．2 つ目は「スマートスライド」と呼ばれる機能である．ユーザは右下に表示された「Next」「Previous」ボタンをクリックしたりタップして，表示されたルートに沿ってビューをスライドすることができる．地図のスケールや表示範囲はスマートズームと同様に，スライド後適切に調整される．これら機能は従来の操作方法の拡張として実装され、従来手法と組み合わせることでより効果的な閲覧を実現する。

提案手法は従来手法に比べ，ユーザの目的とするルートマップ上の情報により簡単かつ少ないインタラクションで，より素早く到達できると考えられる．具体的には 1) ドライブ前にルートの情報をある程度把握しておきたいとき，既知の部分 (自宅周辺など) は飛ばして自信がない部分に素早くアクセスできる，2) ドライブ中に現在地周辺の情報が気になった時，次のいくつかの交差点の概要や目印になるもの，それぞれの交差点まで距離などの情報を素早く得ることができる，3) モバイルデバイスを使用時に片手が塞がっていてマルチタッチジェスチャーが利用できないような時でも，指一本のタップのみによって快適にルートマップを操作することができる，などの利点が考えられる．

アプリケーションはまず対象のルートに対しルートツリーを生成する．スマートズーム機能は，生成されたルートツリーの親子関係をトレースすること

により実現される．スマートスライド機能は，ツリー中の同じ深さのノードをルートの順番に沿ってつなげることにより実現される．木の深さとノードの持つスケール値には強い相関関係があるため，このような単純な手法で関連付けられたノード同士であっても似たようなスケール値を持っていることが多く，実用上はほぼ問題ないと考える．

5.2 TreePrint

TreePrint は，ルートマップを 3 種類のスケールに分けて印刷するためのアプリケーションである．TreePrint は，ルート全体の概要を示すビュー，各 POI に対応する詳細なビュー，そしてその中間のスケールのビューの 3 種類の地図画像を生成する．各地図画像のスケールは，地図に含まれる内容に合わせて適切に調整される．生成した地図画像を用いると，図 8 に示されるような印刷機能を実現することができる．最初のページにはルート全体のビューが印刷され，ルートの各部分に対応する中間スケールのビューがナンバリングされた上地図画像中に示されている．ユーザは各ビューの番号から，ページの下半分に記されたページを参照することにより，該当部分のより詳細な情報を閲覧することができる．

従来の印刷手法と異なり，TreePrint アプリケーションでは，各ビューのスケールはそのビューに含まれる情報に基づき最適化されている．このため従来手法によく見られた 1) 似たようなビューがいくつも生成される，2) ビューのスケールが小さすぎて情報が得られない，などといった問題は発生しない．また，中間スケールのビューの存在により，各 POI 間の位置関係なども明確化され，次の POI までの距離や道の形なども読み取ることができるようになっている．ルート全体が長く，POI が一部に密集している場合などこれらの情報は従来の手法では地図から

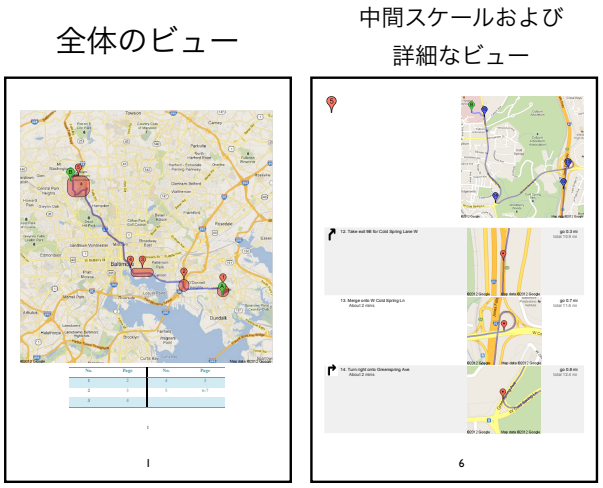

図 8. TreePrint の出力結果を用いた印刷例

ら読み取ることが困難であったが，本手法ではそれを可能にしている．

RouteZoom と同様に，アプリケーションはまず対象のルートに対しルートツリーを生成する．生成されたルートツリーの根が全体のビュー，深さ 1 のノードが中間スケールのビュー，深さ 2 のノードが各 POI の詳細なビューに対応する．長さ 30km 以下の中短距離のルートの場合，多くの場合木の深さは 2 前後で収まる．深さ 3 以上のノードが生成されてしまった場合は，その親をたどって深さ 2 のノードを削除することで木の深さを減らし，最終的な深さを 2 以下に抑える処理を施す．

5.3 評価関数

RouteZoom および TreePrint のアプリケーションの機能を実現するにあたっては，それぞれのアプリケーションに適切なルートツリーを構築する必要がある．これは各アプリケーションに合わせて，3.3 節で説明した最適化に用いるツリーの評価関数を適切に設計することで実現される．注意すべきは，アプリケーションごとに設定するのは評価関数のみでよく，その他のアルゴリズムは同じでよいという点である．これら 2 つのアプリケーション用の評価関数は，共通して以下の式で示される．

$$cost(t) = \sum_{n \in t} \begin{cases} 10^{-15} & (if\ score(n) \leq 0) \\ 1/score(n) & (if\ score(n) > 0) \end{cases}$$

$$score(n) = a \cdot nodesize(n) + b \cdot childsize(n)$$

ただし，t はルートツリー，n はノードを表す．

$nodesize(n)$ はノードの表すビューのサイズに対する制約関数であり，$childsize(n)$ は N の子ノードに対する制約関数である．前者の値域は $[-\infty, 1.0]$，後者は $[0.0, 1.0]$ となっている．a と b は重み付けのパラメタであり，現実装では等しく 0.5 に設定している．RouteZoom においては，$nodesize(n)$ は n の表すビューのサイズが画面サイズの 9 割になったときに最大値を取り，サイズをはみ出した場合ペナルティとして負数を返す．$childsize(n)$ は，n の各子ノード c について，n のビューから見た時に画面サイズの 1/5 になる時に最大値を返す．TreePrint においては，$nodesize(n)$ による制約が画面サイズではなく図 8 に示される各ビューの大きさによる制約になる．また，$childsize(n)$ については子ノードのサイズの基準を 1/16 としている．

6 ユーザ評価実験

RouteZoom アプリケーションについて，16 人の被験者を対象としたユーザ評価実験を行った．実験においては，画面に表示されたルートに関していく

つかユーザに質問をし，RouteZoom および従来のマニュアルな地図操作のいずれかでルートマップを閲覧してその質問に対する答えを探してもらう，というタスクを行なってもらい，従来手法と提案手法の比較を行った．

質問の内容は各交差点の曲がり方や，それぞれの曲がり角の目印になるものなど，地図からの情報の読み取りを問うもので，回答は全て 4 択の選択式となっている．回答はセッションを単位として行われる．各セッションは 4 つの質問セットで構成され，セットごとに 1 つのルートについての 5 項目の質問がある．回答は専用のシステム上で行い，回答までにかかった時間，地図とのインタラクションの回数および時間，問題の正解率などの情報が記録される．

表 1 に，実験結果を簡単にまとめる．t 検定を用いた検証の結果，回答までのインタラクションの数 (クリックやドラッグの回数) やインタラクション時間 (地図の操作にかかった時間．地図の読解時間や回答選択の時間は除く) において，RouteZoom のほうが優位にあることがわかった．

表 1. セッションあたりのインタラクション回数/時間

	RouteZoom	Manual	p (t-test)
回数	70.1 回	136.3 回	.00000359
時間	32.7 秒	55.0 秒	.0000324

実験後，簡単なインタビューを行い，システムについて質的なフィードバックをもらった．ポジティブなコメントとしては,「従来手法より素早く操作できる」「効率の良い操作手法だ」などがあった．一方でネガティブなコメントとしては「慣れるのに時間がかかる」「左右のクリックを間違える」「サジェストされたエリアが何を意味しているのか直観的でない」などがあった．まとめると，提案手法はルートマップの素早い操作という点で優位であるが，若干の習熟コストがかかるということが言える．一方で，ユーザの操作を観察した限りでは，タスクの完了に対して致命的なまでの誤操作や逡巡などは見られなかったため，このコストは 10 分から 30 分程度の練習でクリアできるものであると考える．

7 まとめと今後の課題

本稿では，ルートツリーと呼ばれるルートマップの階層化手法についての提案を行った．ルートツリーは，ルート情報を把握するために必要な地図のビューを自動抽出することで，ユーザのルートに対する理解を促進したり，複数の地図のビューの間を行き来する操作を効率的に補助することができる．さらに本稿では，ルートツリーを自動的に構築するアルゴリズムの提案も行い，ルートマップの閲覧および印刷のための 2 種類のアプリケーションを提案することでその汎用性を示した．

本稿では RouteZoom について評価実験を行ったが，TreePrint についても，実際の利用における可用性を検証するためには，別途評価実験が必要である．また，50km を超える長距離のルートなどで，生成された木が深くなった場合の適切な印刷レイアウトのデザインなども今後の課題となっている．

現時点の実装では，最適化におよそ 15 秒から 30 秒程度の時間を必要とする (最適化を除いた処理コストは 1 秒以下である)．実用性を考えると，最適化の時間コストを少なくとも 5 秒以内に抑える必要があると考える．一方，このようなアルゴリズムの高速化は，データ構造の最適化や，最先端の進化計算アルゴリズム，および並列計算などを用いることによって十分実現可能であると考える．

参考文献

[1] Google Maps. http://maps.google.com/.

[2] MapQuest. http://www.mapquest.com/.

[3] NavEngine. http://developers.cloudmade.com/projects/show/navengine.

[4] M. Agrawala and C. Stolte. Rendering effective route maps. In *Proc. SIGGRAPH '01*, pp. 241–249, 2001.

[5] C. Appert, O. Chapuis, and E. Pietriga. High-precision magnification lenses. In *Proc. CHI '10*, pp. 273–282, 2010.

[6] S. Carpendale, J. Ligh, and E. Pattison. Achieving higher magnification in context. In *Proc. UIST '04*, pp. 71–80, 2004.

[7] G. W. Furnas. Generalized fisheye views. *ACM SIGCHI Bulletin*, 17(4):16–23, 1986.

[8] K. Hornbæk, B. B. Bederson, and C. Plaisant. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM TOCHI*, 9(4):362–389, 2002.

[9] T. Igarashi and K. Hinckley. Speed-dependent automatic zooming for browsing large documents. In *Proc. UIST '00*, pp. 139–148, 2000.

[10] W. Javed, S. Ghani, and N. Elmqvist. Polyzoom: multiscale and multifocus exploration in 2d visual spaces. In *Proc. CHI '12*, pp. 287–296, 2012.

[11] P. Karnick, D. Cline, S. Jeschke, A. Razdan, and P. Wonka. Route visualization using detail lenses. *IEEE TVCG*, 16(2):235–247, 2009.

[12] S. Malacria, E. Lecolinet, and Y. Guiard. Clutch-free panning and integrated pan-zoom control on touch-sensitive surfaces. In *Proc. CHI '10*, pp. 2615–2624, 2010.

[13] D. C. Robbins, E. Cutrell, R. Sarin, and E. Horvitz. ZoneZoom: Map Navigation for Smartphones with Recursive View Segmentation. In *Proc. AVI '04*, pp. 231–234, 2004.