Hierarchical Route Maps for Efficient Navigation

Fangzhou Wang ¹
fzw@acm.org
The University of Tokyo

Yang Li ²
yangli@acm.org
Google Research

Daisuke Sakamoto ¹
d.sakamoto@acm.org
³JST ERATO Igarashi Design Interface Project

Takeo Igarashi ¹,³
takeo@acm.org

¹The University of Tokyo
²Google Research
³JST ERATO Igarashi Design Interface Project

Figure 1: An example of Route Tree. The proposed algorithm automatically extracts meaningful multi-scale views of a route and organizes those views into a hierarchical structure to support efficient navigation. (a) shows the original route and a complete overview of the Route Tree generated from the route, (b) and (c) show close-up view of the Route Tree. Note that these are visualization of the internal data structure and the end-users do not see them directly.

ABSTRACT

One of the difficulties with standard route maps is accessing to multi-scale routing information. The user needs to display maps in both a large scale to see details and a small scale to see an overview, but this requires tedious interaction such as zooming in and out. We propose to use a hierarchical structure for a route map, called a “Route Tree”, to address this problem, and describe an algorithm to automatically construct such a structure. A Route Tree is a hierarchical grouping of all small route segments to allow quick access to meaningful large and small-scale views. We propose two Route Tree applications, “RouteZoom” for interactive map browsing and “TreePrint” for route information printing, to show the applicability and usability of the structure. We conducted a preliminary user study on RouteZoom, and the results showed that RouteZoom significantly lowers the interaction cost for obtaining information from a map compared to a traditional interactive map.

Author Keywords
Route Map Visualization; View Extraction; Multi-scale Navigation.

ACM Classification Keywords
H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces; H.3.3 [Information storage and retrieval]: Information Search and Retrieval. - Retrieval models.

INTRODUCTION

A route map presents route information to guide users from one place to another. Various systems have been developed for automatically generating route maps. These systems have been conventionally implemented in car navigation systems with a local map database. The use of web-based route map generation systems, such as Google Maps [9], has become commonplace for various types of users recently as smart phones and tablet PCs become popular. However, several usability problems still remain with these map systems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
desirable view, which can be very tedious. Therefore, it is desirable to provide an easy way to access various meaningful views with different scales and positions in the route.

We propose to pre-compute a hierarchical structure for a route map (called a “Route Tree”) to provide quick access to such meaningful views (Figure 1). The important features of our hierarchical structure are as follows:

**Feature 1.** Each node represents a part of its parent view in a larger scale, while the root node represents a complete view of the whole route.

**Feature 2.** For every part of the route, there is a node that visualizes the part in the necessary scale.

**Feature 3.** The number of nodes in the tree is made as small as possible, while keeping an appropriate view size (size of the rendered result on the screen or print) for each node.

We present an algorithm for automatically constructing a Route Tree. It starts with an over-segmented tree and iteratively simplifies it by taking balance between compactness and readability. A Route Tree itself is raw tree data, and it can be used in various applications. In this paper, we implemented an interactive map browsing system called RouteZoom to demonstrate the effectiveness of a RouteTree.

Our method is designed to address the situation where a user needs to actively learn or explore an area or route. It is different from but complements automatic uses of maps systems such as Turn-by-Turn navigation used in car navigation in which a user passively receives step-by-step instructions from the system. The contributions of this work are summarized as follows.

1. A novel hierarchical structure for route maps called a RouteTree.
2. A fast algorithm to construct the Route Tree with a few parameters given by the user to control the results.
3. Two useful route map applications that exploit RouteTrees, respectively improved a conventional route map browsing system and a route map printing system.

**RELATED WORK**

**Route map visualization**

Agrawala et al. [1] have proposed a system to render a route map that mimics handwriting maps made by humans. This approach focuses on conveying precise information about POIs and an abstract overview of the complete route. To achieve the goal, it distorts the geographic shape and length of the route and takes away detailed context around the route. However, the lack of contextual information outside the route makes it difficult for users to turn back to the precise route when they mistakenly choose a wrong one, or to determine their precise position in the route map from information such as landmarks around them or the shape of the road. Furthermore, the distorted representation makes it difficult to use the map in combination with advanced services such as aerial photo views.

Several existing web-based map systems [9, 19] are able to print a step-by-step guide showing all POIs and corresponding verbal directions along the route. Karnick et al. [16] proposed a method of putting detailed views of POIs in the route around the complete overview of the route. This can be problematic because the user might want to see the map in some intermediate scales. Furthermore, some of the detailed views are almost the same and therefore redundant because the method shows a detailed view for each POI even in a case where it is possible to show multiple POIs in a single view.

Several previous studies [22, 25] attempted to exploit the user’s prior knowledge about the route in order to produce a more simplified, useful route for users. Other studies [5, 11] propose methods that combine videos or images with conventional route maps to enrich the user’s knowledge about the route. There are also systems that generate Goal-oriented route maps such as tourist maps [10] or Destination-oriented maps [18] rather than Start-to-Goal route maps.

**Hierarchical map structure**

There are several studies on hierarchical structures for map visualization. However, most of them discuss how to automatically make a series of multi-scale map images from geographic data while including information in an appropriate level of detail (e.g., [20, 27]). They do not use them for route map visualization. Studies have been made on hierarchical structures and route maps, but most of them focus not on visualization but on human cognition on routes and its application to generating verbal route directions [17, 26]. Our work differs from these studies in that our method generates a series of maps with appropriate detail in different scales for a route map, and uses them for interactive browsing or compact printing.

**Map browsing interface**

Various methods have been proposed for interactively browsing maps in different scales. Some of them [2, 4] integrated maps of multiple scales into one screen using Focus+Context techniques such as Furnas’ fisheye [7]. Others adopt Overview+Detail techniques to arrange pictures of different levels of focus and size [12, 15]. Another approach is to extend conventional panning and zooming interfaces. Igarashi et al. [13] proposed a method to automatically change zoom level depending on the scrolling speed for map navigation. Zhao et al. [28] proposed to implicitly bookmark user’s view on the map to accelerate browsing speed. Both of the latter two techniques rely only on general user input and do not leverage semantic information associated with routes. Pindat et al. [23] proposed content-aware adaptive fisheye lens that
changes its way of zooming depending on its focus on map. However, the study focuses on interface design and does not discuss in detail how to pre-compute the necessary semantic information it requires from a map.

Robbins et al. [24] proposed a system called “ZoneZoom”, which provides an effective way of navigation on a mobile display. It divides the screen into several areas and allows the user to recursively zoom into them. Though they suggested in their study that their method could be well applied to map navigation, they did not discuss a method to automatically compute zoomable areas. We apply their method as the basis of our user interface for RouteZoom application, but the zoomable areas are automatically computed by our method.

ROUTE TREE CONSTRUCTION
The construction process consists of three steps: 1) Scale Evaluation, 2) Initial Tree Construction, and 3) Optimization (Figure 2). The input route data is taken from a route-search engine (e.g., Google Directions API [8], NavEngine [21]), which includes information such as the road’s geographic shape, the type of road, and POIs. First, we extract a set of “route segments” from the input route data, which represent short, straight lines in the route, and use this information to assign scale values to the segments (Scale Evaluation). Second, we construct an initial tree by hierarchically organizing the route segments depending on their scales (Initial Tree Construction). Since this tree is over-segmented, we optimize the nodes to an appropriate number and size of view to get the final tree (Optimization).

Scale Evaluation
The route shape is initially given as a polyline, and we divide the polyline into short, straight segments with nearly equal lengths. Next, we evaluate a scale value for every route segment. A scale value indicates the minimum scale necessary to present enough information about the segment in the map. For example, motorway segments are visible in a small scale, but small street segments need to be in a larger scale to be visible, because in smaller scale it is usually not rendered on the map. A scale value is an integer ranging from 0 to around 20 (depending on the map) and determines the approximate translation ratio from latitude/longitude coordinates to pixel coordinates in the map as

\[
(x, y) = (f(lat, lon), g(lat, lon))
\]

\[
f(lat, lon) = \frac{2 \cdot \text{scaled}}{360} \cdot (\text{lon} - A)
\]

\[
g(lat, lon) = \frac{2 \cdot \text{scaled}}{360 \cdot \cos(lat)} \cdot (B - \text{lat})
\]

where \(x\) and \(y\) respectively indicate horizontal and vertical coordinates in the map image, and \(\text{lat}\) and \(\text{lon}\) indicate latitude and longitude coordinates. \(A\) and \(B\) respectively indicate longitude and latitude at the top left of the map image [3].

We use the following function to evaluate the scale:

\[
\text{Scale}(R) = \max\{\text{Type}(R), \text{Dist}(R)\}
\]

where \(R\) indicates a route segment.

\(\text{Type}(R)\) returns a scale value of the segment defined by the road type and the type of POI it contains. It depends on the input map data. Table 1 shows the definition for Google Maps. A larger scale value is defined for segments that contain POIs, because those segments usually need a higher scale to be visualized. It returns different values depending on the type of POI (e.g., turning point, branch of roads, continue to a different street), because each of them has a different level of importance for the user and a different scale value for visualization. \(\text{Dist}(R)\) is added in order to prevent neighboring POIs from being too close in the rendered view. It measures the distance between \(R\) and neighboring POIs, and returns a minimum scale value to make the distance larger than a predefined value (we use 30 pixels in our implementation) on the screen.

**Table 1: Example look-up table for Type(R)**

<table>
<thead>
<tr>
<th>Type</th>
<th>NoPOI</th>
<th>Turn</th>
<th>Branch</th>
<th>Continue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motorway</td>
<td>7</td>
<td>11</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>Primary</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>12</td>
</tr>
<tr>
<td>Secondary</td>
<td>10</td>
<td>14</td>
<td>17</td>
<td>13</td>
</tr>
<tr>
<td>Local</td>
<td>14</td>
<td>18</td>
<td>17</td>
<td>17</td>
</tr>
</tbody>
</table>

**Initial Tree Construction**
Figure 3 shows an overview of the method in this step. We first prepare a copy of segmented routes for each scale value within the defined range (Figure 3a). We remove all segments whose scale is smaller than the scale level associated with each copy (Figure 3b). We then group
The returned number of nodes $n$ in tree $t$ is, the less is the greater the returned nodes indicates in its parent view, and vice versa. The size to be similar but randomize the size of area ever param is one against that of the child nodes of $n$. Increasing our implementation. The score $f$ is as follows:

$$
\text{score}(n) = \alpha \cdot \text{nodesize}(n) + \beta \cdot \text{childsize}(n)
$$

where $\alpha$, $\beta$, and $p$ are values ranging from 0.0 to 1.0. $M_{\min}$ and $M_{\max}$ are respectively the minimum and maximum limit of the node size, and $s(n)$ represents the size of node $n$. All these parameters are application-specific i.e. different values are set for different applications. The function sets the ideal node size to $p \cdot M_{\max}$ and gives a negative value as a penalty when the size exceeds its defined limit $M_{\max}$ too much.

The definition of $\text{childsize}(n)$ are as follows:

$$
\text{childsize}(n) = \exp \left( \frac{(M_{\max} / s(n) - m)^2}{2 \cdot \sigma^2} \right)
$$

where $s(n)$ is the size of node $n$ in its parent view. The function sets the ideal child node size to be $1/m$ of the $M_{\max}$ in its parent view, and thus avoids the problem of the area a child node indicates being too large or too small in its parent view.

**GA Optimization**

We use a genetic algorithm (GA) for the optimization. The basic idea is to repeatedly apply the following four operations to the tree in order to reduce the number and increase the size of nodes.

1. Integrate a node to its child nodes (Figure 4a).
2. Integrate a node to its adjacent node (Figure 4b).
3. Increment the scale value of a node (Figure 4c).
4. Add 10 segments (5 for each side) to a node (Figure 4d).

---

Figure 3: Initial tree construction process.

adjacent route segments in each scale level, making nodes (Figure 3c). Each node is associated with a scale level. Finally, an initial Route Tree is created by connecting each node on the basis of the containment relationship among the segments (Figure 3d).

**Optimization**

The initial tree is over-segmented and each node is too small at this point. We therefore optimize the tree by minimizing a cost function to reduce the number and increase the size of nodes.

**Cost Functions**

The definition of the cost function is as follows:

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

where $t$ represents a Route Tree and $n$ represents its node. We set $\alpha=0.5$ and $\beta=0.5$ for weighting parameters $\alpha$, $\beta$ in our implementation. The score function $\text{nodesize}(n)$ is a restriction against the view size of node $n$ and $\text{childsize}(n)$ is one against that of the child nodes of $n$. Increasing parameter $\alpha$ or decreasing $\beta$ will better preserve every node size to be similar but randomize the size of area every child nodes indicates in its parent view, and vice versa. The greater the returned $\text{score}(n)$ value is and the smaller the number of nodes n in tree t is, the less is the $\text{cost}(t)$ that is returned. The definition and graph plot of $\text{nodesize}(n)$ are as follows:

$$
\text{nodesize}(n) = \begin{cases} 
\exp(A \cdot (s(n) - p \cdot M_{\max})) & (s(n) < M_{\min}) \\
B \cdot (s(n) - p \cdot M_{\max})^2 + 1.0 & (M_{\min} \leq s(n) \leq M_{\max}) \\
C \cdot (s(n) - p \cdot M_{\max})^2 + 1.0 & (M_{\max} < s(n)) 
\end{cases}
$$

$$
A = \frac{\log(a)}{M_{\min} - p \cdot M_{\max}} \quad B = \frac{a - 1.0}{(M_{\min} - p \cdot M_{\max})^2} \quad C = \frac{b - 1.0}{(M_{\max} - p \cdot M_{\max})^2}
$$

---

Figure 4: Operations in optimization process.
The algorithm first prepares 500 candidate trees by randomly applying the four operations to the initial tree. It then iteratively reduces the cost of the trees by applying crossover and mutation. The probabilities for crossover and mutation are respectively 0.5 and 0.3. Crossover randomly mixes two selected trees and mutation randomly applies one of the four operations to each node in the selected tree with a probability of 0.1. We use a simple roulette-wheel selection process to make successive generations, and the fitness of each seed is defined as the inverse of the cost evaluated by the Route Tree cost function. After repeating the process 1000 times, the algorithm returns the tree with the lowest score as the final solution.

ROUTE TREE APPLICATIONS
We implemented two Route Tree applications: “RouteZoom” for interactively browsing route maps on a display, and “TreePrint” for printing routing information. We will describe the details of each application respectively in the following two subsections.

RouteZoom
Real-time navigation systems automatically move the map view according to the current position. However, the users may want to browse the map under certain circumstances, e.g., to allow them to 1) memorize directions for parts of the route they are unfamiliar with, and 2) see information for the next few turns to prepare their mind during driving. Therefore, it is still important to improve techniques for browsing route maps for developing better navigation systems.

Figure 5 shows the user interface and functions of RouteZoom. The system has two main functions, the first of which is “smart zoom”. With this function, the system automatically suggests areas in the main map that the user may want to zoom in on to see the details. The user has only to select a suggested area to get a detailed view of it. The user can also get an appropriate zoomed-out view by right-clicking on the map or selecting the (-) command at the bottom right corner. Note that this function not only scales the view but also slides it to efficiently show the route, while in a standard map application the user has to adjust the view position manually after changing the scale. The second function is “smart slide”. With this function, the user can move forward or backward along the route by selecting a command button on the top or bottom of the parent view. The position and scale are automatically adjusted after the slide.

The construction process is as follows. First, we construct a Route Tree from the route. We set construction parameters $M_{max}=800x600px$, $M_{min}=50x50px$, $a=0.1$, $b=0.0$, $p=0.9$, $m=5$, $\sigma=1.0$. Second, we trace the parent-child relationship in the tree to construct the smart-zoom function. Third, we scan the Route Tree from the root to the leaves and connect each POI in the route to nodes that contain it. The smart-slide function gathers nodes connected to the POIs that are next or previous to the current viewport. From the gathered nodes it then chooses one target node that has the nearest scale value to the current view scale. If the difference between the scale value of a target node and that of the current view is less than 3, it simply slides the view to the target node. In case the difference is greater than or equal to 3 we maintain the current scale and slide the view to the next or previous POI’s position.

TreePrint
Although real-time navigation systems are very common these days, there are still strong needs to print out route maps in some situations. We conducted an informal survey to ask 20 daily map users whether they use a printed route map. Surprisingly, 16 of them answered, “Sometimes I may need to print out a route map” and two answered, “I often print out a route map”, while only two answered, “Not necessary at all”. They listed several cases when they needed to print out maps such as traveling abroad in a rented car without a car-navigation system, making a backup for a cell phone low on battery power or a low-accuracy GPS, or just wanting to save money for their cell-phone Internet connection.

The TreePrint system produces a collection of map images for showing route information in three levels of views. It produces a complete overview of a route, detailed views for each POI, and mid-scale views between them. Figure 6a
views with excessively small scale. Furthermore, mid-scale views are indicated on the first page. Mid-scale & POI view

The advantage of enabling a Route Tree to print is that it optimizes the map views depending on the information it contains, while the conventional printing method in a modern map system produces map images with fixed scale regardless of the information they contain. The optimization process avoids the producing of multiple similar views or views with excessively small scale. Furthermore, mid-scale views clarify the spatial relationships between multiple POIs, such as the road shape or the distance between them. This is a clear improvement over conventional systems that have difficulty in visualizing this information when multiple POIs are concentrated in a short part within a long route.

The TreePrint application first produces a Route Tree for the route it prints. The root node of the tree, nodes with depth 1, and nodes with depth 2 respectively represent a complete overview, mid-scale view, and POI-level views. We set construction parameters \( M_{\text{max}} \) and \( M_{\text{min}} \) depending on the depth of the node, which is defined by the map size on each view level. We set the same \( a, b, p, \sigma \) values used to construct RouteZoom, while setting larger \( m \) value \( m=16 \) to reduce the depth of the tree. After Route Tree construction, we set the depth of all leaf nodes to be 2 by iteratively removing parent nodes of leaf nodes whose depths are larger than 2, and adding a child node for leaf nodes whose depths are less than 2. We set the scale value as 15 for newly added children with depth 1 and one greater than its parent node’s scale for depth 2.

RESULTS

Figure 7a shows an example of view transitions in RouteZoom and Figure 7b shows an example of printouts generated by TreePrint. These results are generated from identical route from The Community College of Baltimore to Sinai Hospital of Baltimore.

In Figure 7a, views for which a higher scale is needed to see the details are automatically suggested in each of the views. These views are appropriately connected and the user can use the smart-zoom and smart-slide functions to achieve easy and quick browsing of the route map. The red arrows in the figure suggest zooming transitions between the parent and child views while the blue arrows suggest sliding transitions.

In Figure 7b, the mid-scale view on pages 2-7 effectively visualizes the spatial relationship between each of the POIs, and enables each POI view to have a larger scale to give detailed information without losing context information of the route. The clear hierarchical indexing between complete overview, mid-scale view, and detailed POI view extract views necessary to the user.

EVALUATION

We conducted a user study for RouteZoom by comparing them with conventional methods. RouteZoom is basically an extension of a conventional route map browsing technique, and they both use similar types of map images for presenting information. Therefore, there is essentially no difference in the amount of information they can provide. The difference is in the speed at which correct information can be gotten and the interaction cost needed to get it. Thus, our goal in this user study is to confirm the following two hypotheses:

### Figure 6: Comparison of (a) TreePrint layout (b) Google Maps printing layout. For comparison, POI markers in Google Maps layout overview are manually added. The texts are enlarged from original size for better visual presentation.
H1 The user can gain accurate route information in a shorter time using RouteZoom than using Baseline.

H2 The user can gain accurate route information with a smaller number of interactions and in a shorter interaction time using RouteZoom than using Baseline interface.

To confirm these two hypotheses, we asked participants to work on a route information-searching task to measure the task completion time to operationalize H1, and the number of interactions and the interaction time taken to complete the task to operationalize H2. Note that H1 concerns the overall time for completing a task while H2 only concerns the portion of the task in which the user interacts with the route map. We recruited sixteen participants (twelve males, four females) from an IT corporate environment to conduct this study.

Interfaces
Two interfaces were compared in the study:

RouteZoom: The RouteZoom interface allows the participants to fully use the functions of the application that we described in the previous section.

Baseline: The Baseline interface allows the participants to pan the map by mouse dragging and to zoom by double clicking or scrolling the mouse wheel.

Although more advanced browsing interfaces such as multiview or fisheye exist, we chose a simple one-view panning and zooming interface as the Baseline interface. This is because 1) the Baseline interface is the one having the most basic and popular interaction style among conventional map systems [9, 19] and 2) our method of adding semantic view extraction to navigation interface is orthogonal to advanced methods such as multiview or
fish-eye. They are not mutually exclusive and it is possible to combined with our method. Our study thus complements existing work [6] showing the advantages and disadvantages of such advanced methods.

**Tasks and Apparatus**

The task was to browse a given route map using one of the two interfaces and answer a questionnaire consisting of five item questions about the route information such as “For this turn X, which is the right direction to take”, or “What is the approximate distance from turn X to turn Y?” Figure 8 shows the screenshot of the system we used in the study. The left panel shows the map and the right panel shows the questionnaire. Each questionnaire item has four possible answers. We prepared the questionnaires before inspecting the resulting Rout-Tree generated by the system to avoid potential bias.

Each participant was asked to complete two sessions sequentially, one session using the Baseline interface and the other using RouteZoom. Each session consisted of four trials, for each of which the task was to browse a route map and answer a questionnaire about the route. All participants worked on the same eight questionnaires, but the order and condition assignments were balanced among the participants.

The study was conducted on a Lenovo S20 PC running Ubuntu Linux, with an Intel Xeon 3690 CPU and a 24” monitor with 1920x1200px resolution. The participants used a standard mouse (Logitech G400) with a mouse wheel to interact with the system.

**Measurements**

We measured the performance of each participant by 1) Session completion time, 2) Answer accuracy of the questionnaire, 3) Number of interactions such as clicking or mouse wheel rolling, and 4) Interaction time for operating the map. Note that while the session completion time addresses the total time taken for completing all tasks in the session, the interaction time does not include time for reading maps and answering questions. We also asked participants to answer a questionnaire with 7-point Likert scale and conducted an interview for subjective usability measurements.

**Procedures**

Before starting each session, the participants were instructed to complete two practice trials to familiarize themselves with the system and the browsing interface for the session.

In each trial in the session, the participants began by reading the in-session questionnaire for 30 seconds (the map was hidden during this preparation time) and then they interactively browsed the route map while answering the questionnaire within the next 90 seconds.

After completing each session, the participants were asked to answer a post-session questionnaire about the browsing interface used in the session. Finally, we conducted a post-session interview to collect feedbacks.

**Evaluation Results**

Figure 9a shows the quantitative results for mean session completion time, answer accuracy, number of interactions, and map interaction time taken. The post-session
questionnaire results (7-point Likert scale) are shown in Figure 9b along with the questions asked. We respectively used a paired t-test and a Wilcoxon signed-rank test to analyze the quantitative and qualitative results.

Objective Measurements
There were no significant differences in the session completion time for a session ($p=.979$), meaning that $H_1$ was not supported. On the other hand, there were significant differences in both map interaction time taken ($p=.000$) and number of interactions ($p=.000$), while there were no significant differences in the answer accuracy ($p=.141$). This demonstrates hypothesis $H_2$ was supported.

Table 2 shows detail of interaction time taken and number of interactions per session for each interface. The results show that more zoom interactions are replaced by automatic zooming ($(z_r-z_{th})/z_{th}=0.70$ in time and 0.68 in number) than pan interactions by automatic panning ($(p_r/p_{th})=0.50$ and 0.57). On the other hand, the replaced interaction cost is reduced more efficiently by automatic panning $(p_r/(p_r+p_{th})=0.20$ in time and 0.12 in number) than automatic zooming $(z_r/(z_r-z_{th})=0.42$ and 0.31).

One possible reason $H_1$ was not supported is that the user's behavior was mostly affected by the given time constraint (90 sec per question set) and not by the technique. The users apparently used the extra time due to the reduced interaction time to work on problems longer. This did not contribute to improved answer accuracy because most of the mistakes were the result of fundamental misunderstandings about the route information.

Although $H_1$ was not supported in our study, the reduced number of interactions and interaction time by RouteZoom would have considerable impact on navigation safety, because less interaction cost will greatly help to prevent the user from being distracted from their driving or walking. It may also have a positive impact on the session completion time in terms of practical conditions, because many car navigation systems adopt more adverse input conditions than those in our user study (high performance wheel mouse and large display).

Subjective Evaluations
The post-session questionnaire results showed that on the whole the participants preferred RouteZoom to the Baseline interface (Figure 9b). In particular, they felt RouteZoom was a “quicker” way to browse the route maps than the Baseline method. This is probably because the users had to constantly move their hands and fingers to change the view with the Baseline method and thus felt the time taken for the interaction was the bottleneck. RouteZoom removed this bottleneck and was positively received by the users.

The results also show that the users felt it was easier to browse route maps with fewer operations when using RouteZoom. This corresponds to the reduced interaction-cost shown in the quantitative results. With respect to the learning effort needed for using these two methods, the results for Q4 show that although the participants were already familiar with the conventional method, for them there was no significant difference between the two methods. This indicates that RouteZoom is easy to learn. The participants also indicated they would like to use RouteZoom in the future.

In addition to answering the structured post-session questionnaire, participants were asked to give comments and suggestions about RouteZoom in the post-study interview. We received many positive comments, such as that RouteZoom was “quick and easy” and that it provided “good positioning to appropriate views.” They also pointed out usability issues that we can improve on, such as “Sometimes suggestions in the map were too small to read easily.” It is worth mentioning that while several participants highly preferred the smart slide function, several others never used it during the study and commented that they did not want to shift their focus from the map to the command button at the bottom left corner. We can easily address this issue by providing a keyboard shortcut for triggering the function.

DESIGN IMPLICATIONS
A Route Tree is a general representation with various possible applications, which is not limited to two applications we proposed in this paper. For example, advanced browsing techniques such as multiview control and fisheye zooming can be applied to a Route Tree. We can enable users to select suggested areas on the parent view to instantly change not only the position but also the zoom level of the detailed view. By enabling the Route Tree to support the fisheye technique, we can allow the system users to automatically change the zoom level depending on the position of the fisheye lens on the map.

A Route Tree also enables the user to control the route map by giving voice commands such as “Zoom in to area 1”, “Zoom out”, or “(Slide to) next”. This would be much faster and less distracting for the users than existing voice-enhanced browsing methods (e.g., [14]), because most of them require the user to constantly focus on the display.

A Route Tree could also improve the automatic view transition function provided by real-time step-by-step navigation system. It would reduce the number of view
transitions by optimizing the number of views to a minimum while preserving necessary information for the users. This would provide a less stressful navigation experience to the user.

CONCLUSION
We proposed a hierarchical structure for route maps called a “Route Tree.” It helps users to move back and forth across multiple views with different scale levels and positions in a map. We also proposed an algorithm to automatically construct a Route Tree and two Route Tree applications. These are “RouteZoom”, an interactive map application where the user interactively browses a hierarchical map, and “TreePrint”, which uses the hierarchical structure to generate 3 kinds of map images suitable for printing out. We conducted a user study with RouteZoom. The results showed that the application effectively reduces users’ map interaction costs and is well perceived by the user in terms of its quickness and easiness to use.

REFERENCES