Rendering Effective Route Maps: Improving Usability Through Generalization

Maneesh Agrawala Chris Stolte
Stanford University*

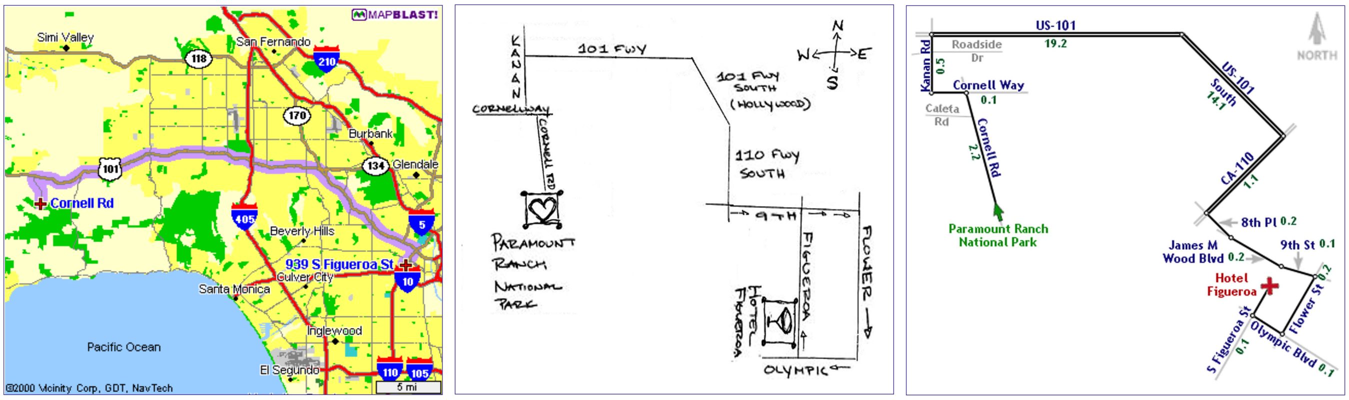

Figure 1: Three route maps for the same route rendered by (left) a standard computer-mapping system, (middle) a person, and (right) LineDrive, our route map rendering system. The standard computer-generated map is difficult to use because its large, constant scale factor causes the short roads to vanish and because it is cluttered with extraneous details such as city names, parks, and roads that are far away from the route. Both the handdrawn map and the LineDrive map exaggerate the lengths of the short roads to ensure their visibility while maintainaing a simple, clean design that emphasizes the most essential information for following the route. Note that the handdrawn map was created without seeing either the standard computer-generated map or the LineDrive map. *(Handdrawn map courtesy of Mia Trachinger.)*

Abstract

Route maps, which depict a path from one location to another, have emerged as one of the most popular applications on the Web. Current computer-generated route maps, however, are often very difficult to use. In this paper we present a set of cartographic generalization techniques specifically designed to improve the usability of route maps. Our generalization techniques are based both on cognitive psychology research studying how route maps are used and on an analysis of the generalizations commonly found in handdrawn route maps. We describe algorithmic implementations of these generalization techniques within LineDrive, a real-time system for automatically designing and rendering route maps. Feedback from over 2200 users indicates that almost all believe LineDrive maps are preferable to using standard computer-generated route maps alone.

Keywords: Information Visualization, Non-Realistic Rendering, WWW Applications, Human Factors

1 Introduction

Route maps, which depict a path from one location to another, are one of the most common forms of graphic communication. Although creating a route map may seem to be a straightforward task, the underlying design of most route maps is quite complex. Mapmakers use a variety of cartographic generalization techniques including distortion, simplification, and abstraction to improve the clarity of the map and to emphasize the most important information [16, 21]. This type of generalization, performed either consciously or sub-consciously, is prevalent both in quickly sketched maps and in professionally designed route maps that appear in print advertisements, invitations, and subway schedules [25, 13].

Recently, route maps in the form of driving directions have become widely available through the Web. In contrast to hand-designed route maps, these computer-generated route maps are often more precise and contain more information. Yet these maps are more difficult to use. The main shortcoming of current systems for automatically generating route maps is that they do not distinguish between essential and extraneous information, and as a result, cannot apply the generalizations used in hand-designed maps to emphasize the information needed to follow the route.

Figure 1 shows several problems arising from the lack of differentiation between necessary and unnecessary information. The primary problem is that current computer-mapping systems maintain a constant scale factor for the entire map. For many routes, the lengths of roads can vary over several orders of magnitude, from tens of feet within a neighborhood to hundreds of miles along a highway. When a constant scale factor is used for these routes, it forces the shorter roads to shrink to a point and essentially vanish. This can be particularly problematic near the origin and destination of the route where many quick turns are often required to enter or exit a neighborhood. Even though precisely scaled roads might help navigators judge how far they must travel along a road, it is far more important that all roads and turning points are visible. Handdrawn maps make this distinction and exaggerate the lengths of shorter roads to ensure they are visible.

Another problem with computer-generated maps is that they are often cluttered with information irrelevant to navigation. This extraneous information, such as the names and locations of cities, parks, and roads far away from the route, often hides or masks information that is essential for following the route. The clutter makes the maps very difficult to read, especially while driving. Handdrawn maps usually include only the most essential information and are very simple and clean. This can be seen in figure 1(middle) where even the shape of the roads has been distorted and simplified to improve the readability of the map. Furthermore, distorting

*(maneesh,cstolte)@graphics.stanford.edu

the lengths of shorter roads and removing unnecessary information makes it possible to include and emphasize helpful navigational aids such as major cross-streets or landmarks before the turns.

Despite the fact that the distortion techniques used in hand-designed maps improve usability, there has been surprisingly little work on developing automatic cartographic generalization techniques based on these distortions. Existing research on automatic generalization has focused on developing simplification and abstraction techniques for standard road, geographical, and political maps [4, 16, 14]. Unlike route maps, these general purpose maps are designed to convey information about an entire region without any particular focus area. Therefore, these maps cannot include the specific types of distortion that are used in route maps.

This paper presents two main contributions:

Route Map Generalization Techniques: We have developed a set of generalization techniques specifically designed to improve route map usability. Our techniques are based on cognitive psychology research showing that an effective route map must clearly communicate all the turning points on the route [6], and that precisely depicting the exact length, angle, and shape of each road is much less important [28]. We consider how these techniques are applied in handdrawn maps and show that by carefully distorting road lengths and angles and simplifying road shape, it is possible to clearly and concisely present all the turning points along the route.

Automatic Route Map Design System: We describe LineDrive, an automatic system for designing and rendering route maps. LineDrive takes advantage of our route map generalization techniques to produce maps that are much more usable than those produced by standard computer-based map rendering systems. LineDrive performs a focused randomized search over the large space of possible map designs to quickly find a near-optimal layout for the roads, labels, and context information. An example of a LineDrive map is shown in figure 1(right). Feedback from over 2200 users indicates that almost all believe LineDrive maps are preferable to using standard computer-generated route maps alone.

In computer graphics we usually consider distortion and abstraction techniques within the area of non-photorealistic rendering. To apply these techniques to visualization requires understanding how the techniques can improve the perception, cognition, and communicative intent of an image. Earlier examples of this approach to visualization include Mackinlay's [17] investigation of methods for automating chart and graph design, Seligmann and Feiner's [23] research on the automatic design of intent-based illustrations, and Interrante's [15] work on using illustration techniques to improve the perception of 3D surface shape in volume data. In this paper we extend this same approach to the automatic design of route maps.

The remainder of this paper is organized as follows. In section 2, we examine the specific generalization techniques applied in handdrawn route maps and how these techniques improve map usability. Section 3 describes algorithmic implementations of these techniques in LineDrive. Results are presented in section 4, and section 5 discusses conclusions and future work.

2 Route Map Design

In order to design a better route map, we begin by analyzing the tasks involved in following a route. From this analysis, we identify the essential information a route map must communicate to support these tasks. We then describe how we use specific generalization techniques, including distortion and abstraction, to present this information in a clear, concise, and convenient form.

2.1 Information Conveyed by Route Maps

Understanding how people think about and communicate routes can provide great insight into what information should be emphasized in a computer-generated route map. A common theory in the field of cognitive psychology is that people think of routes as a sequence of turns [27, 16]. It has been shown that verbal route directions are generally structured as a series of turns from one road to the next and that emphasis is placed on communicating turn directions and the names of the roads [7]. Tversky and Lee [28] have shown that handdrawn maps maintain a similar structure with emphasis on communicating the roads and turn direction at each turning point.

A turning point can be defined by a pair of roads (the road entering and the road exiting the turning point) and the turn direction (left or right) between those two roads. Route maps depict this information visually, so navigators can quickly scan the map to find the road they are currently following and look ahead to determine the name of the next road they will turn onto. Once the name of the next road is known, the navigator can search for the corresponding road in the physical world. The turn direction specifies the action navigators must take at the turning point.

Although it is possible to follow a route map that only indicates the road names and turn direction at each turning point, additional information can greatly facilitate navigation and is often included in hand-designed maps. For example, if the map labels each road with the distance to be travelled along that road, navigators can use their odometer to determine how close they are to the next turn. Cross-streets and local landmarks along the route, such as buildings, bridges, rivers, and railroad tracks, can also be used for gauging progress. Navigators can also use this information to verify that they are still following the route and did not miss a turn. However, cross-streets and local landmarks are not essential for following the route and are usually included in the map only when they do not interfere with the primary turning point information.

2.2 Generalizing Route Maps

Although route maps may be used before a trip for planning purposes, they are most commonly used while actually traversing the route. In many cases, navigators are also drivers and their attention is divided between many tasks. As a result, they can only take quick glances at the map. Therefore, maps must convey the turning point information in a clear, easy-to-read manner and must have a form-factor that is convenient to carry and manipulate.

Most current styles of route maps fail these requirements. A common approach to route mapping is to highlight a route on a standard road map that uses a constant scale factor and depicts all the roads within a region. This style is used by current computer-based route map rendering systems and, as shown in figure 1(left), the constant scale factor makes it impossible to see short roads and their associated turning points. Strip maps, or triptiks, address the issue of varying scale by breaking the route up onto several maps, each with its own orientation and scale. However, the changing orientation and scale make it difficult to understand the overall layout of the route and how the different maps correspond to one another.

One existing route mapping style, the handdrawn map, manages to display each turning point along the route clearly and simultaneously maintain simplicity and a convenient form factor. This is accomplished by performing three types of generalization on the route: (1) the lengths of roads are distorted, (2) the angles at turning points are altered, and (3) the shapes of the individual roads are simplified. We consider each of these in turn:

Length Generalization: Handdrawn maps often exaggerate the lengths of shorter roads on the route while shortening longer roads to ensure that all the roads and the turning points between them are visible, as shown in figure 1. This distortion allows routes containing roads that vary over several orders of magnitude to fit within a conveniently sized image (i.e. a single small sheet of paper). The distortion is usually performed in a controlled manner so that shorter roads remain perceptually shorter than longer roads, while maintaining the overall shape of the route as much as possible.

Angle Generalization: Mapmakers often alter the angles of

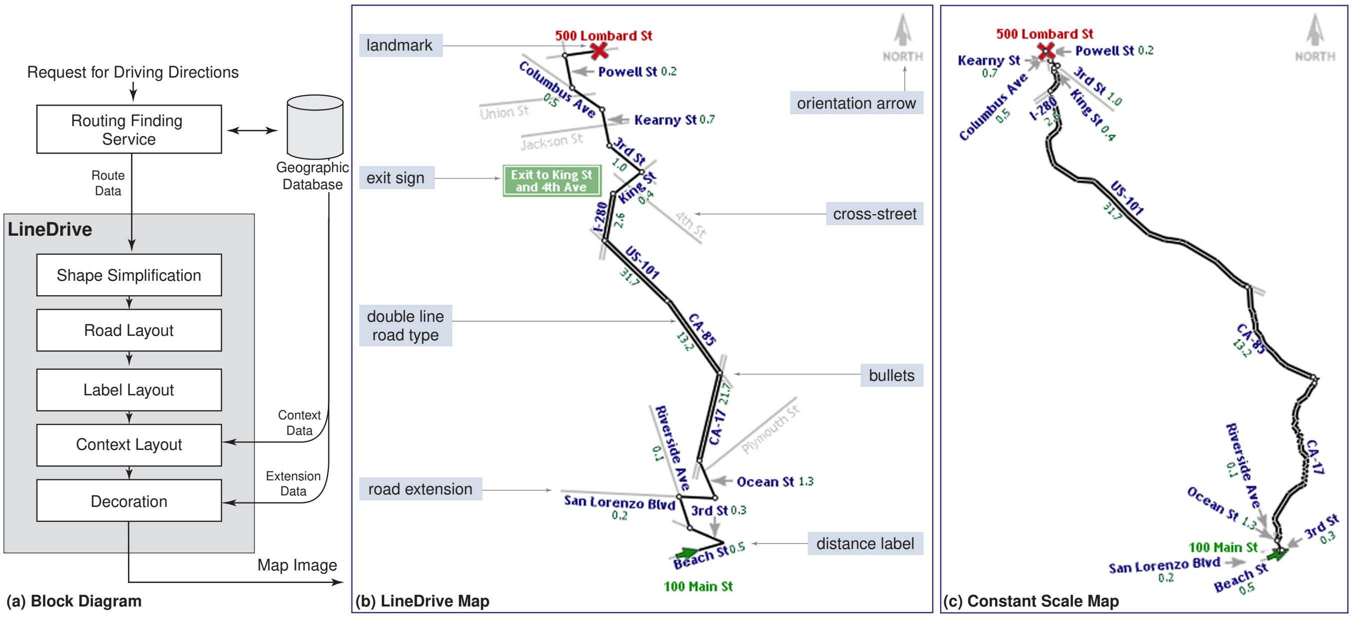

Figure 2: The LineDrive system. (a) Given a route as a sequence of roads, LineDrive designs a route map by processing the route through five consecutive stages. (b) The resulting LineDrive map. (c) The same map rendered without applying the generalization techniques performed by LineDrive. The constant scale factor and retention of detailed road shape make it difficult to identify many of the roads.

turns to improve the clarity of the turning points. Very tight angles are opened up to provide more space for growing shorter roads and labeling roads clearly. Roads are often aligned with the horizontal or vertical axes of the image viewport, to form a cleaner looking, regularized map [26]. Such angular distortions are acceptable because reorienting correctly requires knowing only the turn direction (left or right), not the exact turning angle.

Shape Generalization: Since a navigator does not need to make active decisions when following individual roads, knowing the exact shape of a road is usually not important. Simplifying the road shape removes extraneous information and places more emphasis on the turning points, where decisions need to be made. Roads with simplified shape are perceptually easier to differentiate as separate entities and are also easier to label clearly.

While these generalization techniques can increase the usability of the route map, they can also cause confusion and mislead the navigator if carried to an extreme. By simplifying road shape and distorting road lengths and angles, it is possible to drastically change the topology and overall shape of the route. When these generalizations are performed carefully, however, they can dramatically improve the usability of the map.

3 System

The LineDrive system automatically designs route maps in real-time using the generalization techniques commonly found in hand-drawn maps. The space of all possible route map designs and layouts is extremely large and contains many dimensions. We reduce the dimensionality of this space by performing the map design in five independent stages as shown in figure 2.

All of the geographic data is stored in a database in the standard latitude/longitude geographical coordinate system. The route finding service computes the sequence of roads required to get from a given origin to a given destination and passes this sequence into LineDrive. Each road is represented as a piecewise linear curve described by a sequence of latitude/longitude shape points.

The first stage of LineDrive is shape simplification, which removes extraneous shape detail from the roads, as described in section 3.1. The next three stages, road layout, label layout, and context layout, each deal with automating a layout problem. We use a similar search-based approach in all three stages, described in section 3.2. The details of each layout stage are then presented in sections 3.3 through 3.5. The decoration stage, described in section 3.6, adds elements such as road extensions and an orientation arrow to the map to enhance its overall usability. We conclude our system description in section 3.7 by presenting methods for computing image size based on the aspect-ratio of the route and the size of the output display device. Our system description provides an overview of how we automate the route map design process. Further system implementation details can be found in [1].

3.1 Shape Simplification

LineDrive's shape simplification stage reduces the number of segments in each road while leaving the overall shape of the route intact. Shape simplification not only yields a cleaner looking map but also increases the speed and memory efficiency of the subsequent layout stages of the system.

Techniques for curve smoothing, interpolation, and simplification have been well-studied in a variety of contexts including cartography [22, 8, 2] and computer graphics [10, 12, 5]. We take a standard approach to simplification that ranks the relevance of all the shape points of the curve and then removes all interior shape points that fall below a given threshold. However, our simplification algorithm must not introduce the three undesirable effects shown in the rightmost column of figure 3: false intersections, missing intersections and inconsistent turn directions.

We include three tests during simplification to prevent these problems. To ensure that the simplification process does not introduce false or missing intersections, we initially compute all the *true* intersection points between each pair of roads. Suppose roads r_1 and r_2 initially intersect at point p_1. We add the intersection point p_1 to the set of shape points for both r_1 and r_2 and mark p_1 as *unremovable*. Since the simplification algorithm cannot remove these unremovable intersection points, a missing intersection cannot be generated. Moreover, we only accept the removal of a shape point as long as its removal does not create a *new* intersection point that is not in our original list of true intersection points. This test ensures that the simplification will not introduce any false intersections. Finally, we check for inconsistent turn direction at the turning

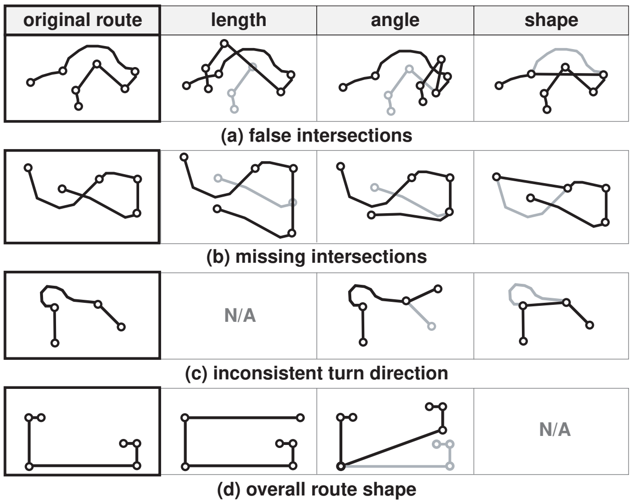

Figure 3: Generalization can cause four types of undesirable effects. Each column shows the route after generalizing the length, angle, or shape of a single road. For comparison, the undistorted route is shown in gray. (a) The original route does not contain an intersection but generalization causes false intersections. (b) The original route contains an intersection (this usually occurs when one road passes over another road) but after generalization the intersection is missing. (c) Generalization causes a right turn to appear as left turn or vice versa. Note that distorting road length cannot generate an inconsistent turn direction. (d) Generalization causes drastic changes in overall route shape. This is reflected in substantial changes in the length and direction of the vector between the route endpoints. Our shape simplification algorithm cannot cause drastic changes to the overall route shape because it only removes shape points from each road and never removes the first or last shape point of a road.

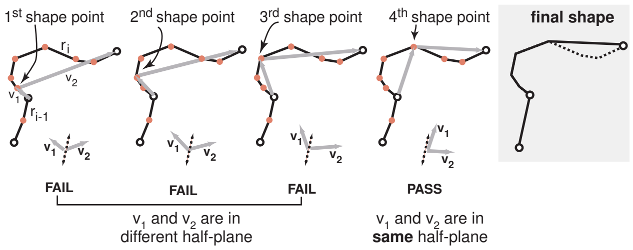

Figure 4: Turn direction consistency check between roads r_i and r_{i-1}. We step through the shape points of r_i, forming two vectors: v_1, between the endpoint of r_{i-1} and the current shape point, and v_2, between the current shape point and the endpoint of r_i. If v_1 and v_2 are not in the same half-plane with respect to the coordinate system oriented along the last segment of the r_{i-1}, we mark the current shape point as unremovable. The test continues until a shape point is not marked as unremovable.

points between each road r_i and the roads r_{i-1} and r_{i+1} adjacent to it. We describe the test between r_i and r_{i-1} in figure 4. The test between r_i and r_{i+1} is similar.

For most roads we are very aggressive about simplification. We remove all shape points that are not marked as unremovable by the previous tests, so most roads are simplified to a single line segment. For some roads, such as highway on- and off-ramps, depicting more realistic shape can be useful. Knowing whether a ramp curves around tightly to form a cloverleaf or only bends slightly can make it easier to enter or exit the highway. Thus, when simplifying ramps we use a more conservative simplification relevance metric to retain more shape [2].

Some long routes between distant cities require traversing many highways. Depicting all the short ramps between the highways can clutter the map with unnecessary detail. Therefore, if the route is longer than a given threshold we remove all ramps from the map that can be removed without creating a false or missing intersection or inconsistent turn direction. Note that all the ramps have been removed from the map in figure 2(b).

3.2 Formulating Layout As Search

In almost any layout problem there are constraints on how the information can be laid out, and there are a set of criteria that can be used to evaluate the quality of the layout. Many such layout problems can be posed as a search for an optimal layout over a space of possible layouts. To frame the layout problem as a search we need to define an initial layout and two functions: a *score* function that assesses the quality of a layout based on the evaluation criteria, and a *perturb* function that manipulates a given layout to produce a new layout within the search space. We can then perform simulated annealing [20] to search for a layout that minimizes the score, as shown in the following pseudo-code:

procedure SimAnneal()
InitializeLayout()
$E \leftarrow$ ScoreLayout()
while(! termination condition)
  PerturbLayout()
  $newE \leftarrow$ ScoreLayout()
  if $((newE > E)$ and (Random() $< (1.0 - e^{-\Delta E/T})))$
    RevertLayout()
  else
    $E \leftarrow newE$
  Decrease(T)

The simulated annealing algorithm accepts all good moves within the search space and, with a probability that is an exponential function of a temperature T, accepts some bad moves as well. As the algorithm progresses, T is annealed (or decreased), resulting in a decreasing probability of accepting bad moves. Accepting bad moves in this manner allows the algorithm to escape local minima in the score function.

The difficult aspects of characterizing the layout problem as a search are designing an efficient score function that captures all of the desirable features of the optimal layout and defining a perturb function that covers a significant portion of the search space. As we discuss the different layout stages of LineDrive, we will focus on explaining these aspects of our algorithm design.

3.3 Road Layout

The goal of road layout is to determine a length and an orientation for each road such that all roads are visible and the entire map image fits within a pre-specified image size. Moreover, the layout must avoid the problems shown in the second and third columns of figure 3 and preserve the topology and overall shape of the route.

To generate an initial layout for the search, we first build an axis-aligned bounding box for the original route and compute a single factor to scale the entire route to fit within the given image viewport. Next, we grow all roads that are shorter than a predefined minimum pixel length, L_{min}, to be L_{min} pixels long. Since we initially scaled all the roads to fit exactly within the bounds of the image, growing the short roads may extend the map outside the viewport. We finish the initial layout phase by again scaling the entire route to fit within the image viewport.

To perturb a road layout during the search, we randomly choose a road r_i and either scale its length $l(r_i)$ by a random factor between $0.8x$ and $1.2x$, or change its orientation by a random reorientation angle between ± 5 degrees. The ± 5 degree bound on road reorientation is decreased as necessary to ensure that an inconsistent turn direction is not introduced. After modifying a road, we rescale the route to fit within the image viewport. By disallowing perturbations that cause inconsistent turn directions and forcing the route to always fit the viewport, we limit our search space to maps that meet our turn direction and image size constraints.

All other constraints on road layout are enforced through the scoring function which examines three aspects of the road layout: road length and orientation, intersections between roads, and the shape of the overall route. We discuss the computation of these component scores in the next three subsections. After the road layout search is complete, we fine-tune the road orientations as described in section 3.3.4.

3.3.1 Road Length and Orientation

Each road r_i is scored on two length-based criteria. First, we penalize any road that is shorter than L_{min} using the following formula:

$$score(r_i) = ((l(r_i) - L_{min})/L_{min})^2 * W_{small} \quad (1)$$

where W_{small} is a predefined constant used to control the weight of the score in relation to the other scoring criteria[1]. The function is quadratic rather than linear, so roads that are much shorter than L_{min} are given a higher penalty than roads that are just a little shorter than L_{min}. Recall that simulated annealing decides whether to accept the current layout based on the difference between the current score and the previous score. By using a quadratic function, we increase the probability of accepting perturbations which grow the shortest roads because such perturbations yield the largest change in score per pixel length. If we used a linear function, growing any road by an amount x would yield the same change in score with no preference for growing the shortest roads.

The second length-based scoring criterion considers the relative ordering of the roads by length. We add a constant penalty for each pair of roads whose length ordering has shuffled between the original map and the current layout. The purpose of this score is to encourage layouts in which the longer roads *appear* longer than shorter roads in the final map. Therefore, we only consider roads as being shuffled when the difference in their lengths is greater than a predefined perceptual threshold (usually 5-10 pixels).

We also penalize each road by a score proportional to the difference between its current orientation α_{curr} and its original orientation α_{orig} using the following formula:

$$score(r_i) = |\alpha_{curr} - \alpha_{orig}| * W_{orient} \quad (2)$$

Since this score is minimized when the current orientation is closest to its original orientation, we only introduce substantial changes to road orientation if the change helps minimize some other component of the road layout score. For example, a substantial change in orientation may be introduced to resolve a false intersection.

3.3.2 Intersections

Both missing and false intersections can be extremely misleading, so we severely penalize any proposed layout containing these problems. We first describe how simple missing and false intersections are resolved independently and then describe how scoring must change when a layout contains both missing and false intersections.

Simple Missing Intersections: There are two forms of missing intersections. A true *missing* intersection occurs when two roads should intersect, such as when a highway ramp loops over or under the highway, but don't. A *misplaced* intersection occurs when two roads should intersect and do, but at the wrong point. As shown in figure 5, in both cases we compute a score that is proportional to the Euclidean distance between the proper intersection point on each road. However, since it is more important for the proper pair of roads to intersect than it is for the point of intersection to be placed exactly, we set the scoring weight for a misplaced intersection to be much lower than for a missing intersection.

Simple False Intersections: False intersections occur when the path incorrectly folds back on itself, forming a loop or knot. One way to remove an individual knot is to move the route endpoint

[1] Each of our component scores uses a similar weighting constant.

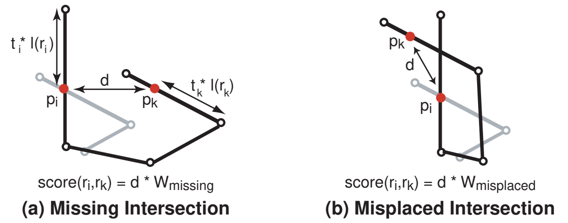

Figure 5: Scoring missing and misplaced intersections. In both cases the score is proportional to d, the Euclidean distance between the two points p_i and p_k that should intersect (marked in red). Initially for each pair of intersecting roads r_i and r_k we compute the parametric values t_i and t_k of the intersection point. Multiplying these parameters by the current lengths of the roads $l(r_i)$ and $l(r_k)$ gives us the current position of p_i and p_k. For comparison, the original route is shown in gray.

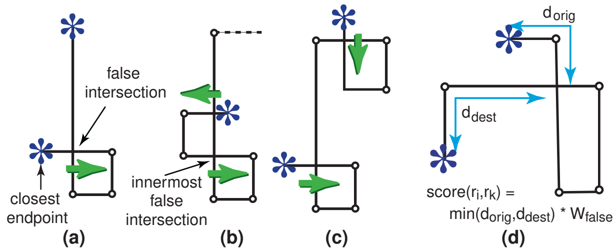

Figure 6: Handling false intersections. (a),(b),(c) The direction the route endpoints should move to independently resolve each false intersection is indicated by the large green arrows. (b) The two false intersections pull the endpoint in opposite directions. This is addressed by counting only the innermost false intersection score. (c) The innermost false intersection is scored for each endpoint independently, so in this case both false intersections are included in the final score. (d) The score for a simple false intersection is proportional to the distance to the closest endpoint of the route as measured in pixels *along the route.*

closest to the intersection (measured in pixels along the route) towards the intersection point. Figure 6(a)-(c) illustrates several false intersection scenarios, showing for each intersection point which direction the closest endpoint must move to remove the knot.

For each false intersection we compute a score proportional to the distance in pixels along the route to the nearest endpoint, as shown in figure 6(d). This approach is conceptually equivalent to building a scoring hill along the route that guides the closest endpoint towards the intersection point, thereby unravelling the knot.

When a route contains multiple false intersections, the false intersection scores may conflict and push the endpoint in opposite directions, as shown in figure 6(b). We address this problem by counting only the score for the innermost false intersection (working inwards from the endpoint to the center of the route). By penalizing the layout for only the innermost false intersection, we guide the endpoint towards the desired direction and eventually resolve both false intersections.

False Intersections and Missing Intersections: In most cases when false and missing intersections occur in the same map, the scores interact properly to resolve both problems. There is one exceptional situation that occurs when the loop formed by a false intersection contains a missing intersection. As shown in figure 7, one score may push in one direction and the other score in the other direction, resulting in a stalemate in which neither problem can be resolved. In both of these cases there is supposed to be an intersection; it is just occurring between the wrong roads. We resolve the situation with an additional rule: if either point of a missing intersection is inside the loop formed by a false intersection, we add a constant penalty for the false intersection, rather than a hill-based score. Using a constant false intersection score allows the missing intersection score to guide the intersection to the desired location.

Extended Intersections: While the false and missing intersec-

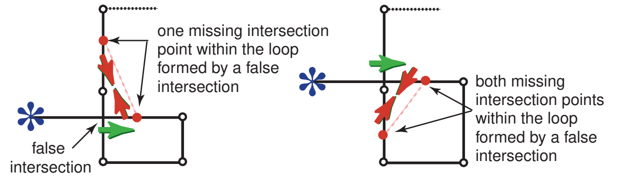

Figure 7: Interactions between false and missing intersections. In both cases, the false and missing intersection scores push points on the route in conflicting directions, as indicated by the arrows. To resolve the conflict, we add a constant penalty for the false intersection and allow the missing intersection score to pull the intersection to the desired location.

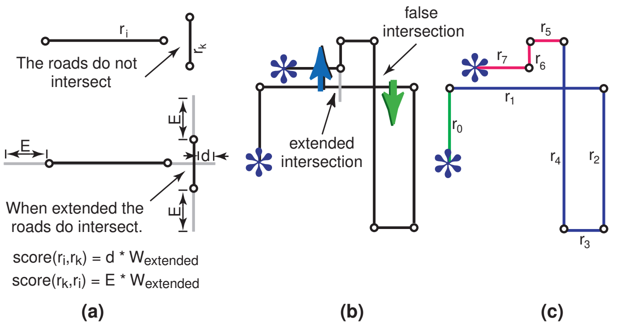

Figure 8: (a) Scoring extended intersections. (b) The extended intersection and false intersection scores conflict and push the layout in opposite directions. (c) All roads between a route endpoint and a false intersection or between a pair of false intersections are considered to be in the same *false intersection interval*. In this case, there are three intervals $[r_0]$, $[r_1, r_2, r_3, r_4]$, and $[r_5, r_6, r_7]$. We resolve the conflict by only counting extended scores between roads in the same false intersection interval. Since r_1 and r_6 are in different intervals, their extended intersection score is not counted.

tions scores are essential for maintaining the overall topology of the route, they do not consider the spacing between roads. It is possible for the perturb function to generate road layouts in which non-intersecting roads pass so close to one another that they incorrectly appear to touch. We identify such layouts by checking for *extended intersections* between each pair of roads. We extend the endpoints of each road by a fixed pixel length E and then check if the resulting roads intersect.

Extended intersections are scored as shown in figure 8(a). If the intersection occurs on the extended portion of the road as for r_i, the score is proportional to the distance between the intersection point and the extended endpoint of the road. If the intersection occurs within the main extent of the road as for r_k, the score is set to the largest possible penalty for intersection with the extended portion of the road. As shown in figure 8(b), it is possible for an extended intersection score to conflict with a false intersection score. To reduce such conflicts, we include extended intersection scores only when the extended intersection occurs between two roads in the same *false intersection interval*, as shown in figure 8(c).

3.3.3 Route Shape

The final road layout score considers the overall shape of the route. As shown in figure 3, perturbing the lengths and angles of each road can drastically alter the overall shape. It is possible for a destination that should appear to the west of the origin to end up appearing to the east of the origin, and the origin can sometimes appear much closer to the destination than it actually is.

To reduce such problems, we compute two road layout scores based on the vector between the origin and destination of the route. The endpoint direction score penalizes layouts that alter the direction of this vector and is proportional to the difference in angle between this vector in the original map and in the current map. The endpoint distance score penalizes layouts in which distance between the origin and destination is smaller than a minimum length based on the original distance between them.

3.3.4 Fine-Tuning Road Orientation

Once the search phase of road layout is complete, we snap each shallow angle road in the final layout to the nearest horizontal or vertical axis. Roads that form shallow angles (i.e. < 15 degrees) with the image plane horizontal or vertical axes tend to increase the visual complexity of the map. Furthermore, such roads can be difficult to antialias, especially on personal digital assistant (PDA) displays with limited color support. Note that we only reorient a road if doing so does not introduce an inconsistent turn direction or a false, missing, or extended intersection.

3.4 Label Layout

For the route map to be usable, each road on the map must be labeled with its name. Similarly, the origin and destination of the route should be labeled with their addresses. Each label is added to the map to communicate a piece of information (e.g. a road's name) through a combination of text and images. The label's placement and style further communicate which map object (e.g. road, landmark, etc.) it is labeling. We refer to this object as the label target.

There are many different ways to label a given target object. A typical method for labeling roads is to simply write the name directly above or below the road. This approach uses proximity to associate the label with its target road. Another style is to put the text near the road and then add an arrow pointing to the road to form the association between the name and its target. Figure 9 shows several styles that might be used to label different objects. As shown in figure 10, a *labeling style* is comprised of three components:

- **Graphic Elements:** A set of text and image elements. The primary graphic element is usually a name, and secondary graphic elements can include distance to travel, arrows, highway shields, etc.
- **Arrangement:** The arrangement of the secondary graphic elements relative to the primary element. For example, the arrow-left-of-name labeling style puts the arrow graphic to the left of the primary name graphic.
- **Placement Constraints:** Each constraint is a region in the map image defining a set of valid positions and orientations for the center of the primary graphic.

To place a given label in the map, we must choose both a labeling style and a label location from within one of the placement constraint regions for that style. Therefore, our label layout search space is defined by the set of possible labeling styles and the placement constraints for each style, for every label in the map.

In the first phase of label layout, we create a list of possible labeling styles for each target object by considering factors such as the size, shape, and type of the target (e.g. highway, residential road, or landmark) and the length of the label name (e.g. if the name is long we might create a word-wrapping style). Each style is also given a rank based on its desirability. For example, for roads, the *along-road* style is preferable to to the *arrow-left-of-name* style.

We create an initial label layout by placing each label at the most central position within its highest ranked labeling style. We then deterministically fix as many labels as possible. We check if each label in its initial position could ever conflict with the placement of any other label by intersecting each label in its initial position with all potential positions for every other label. The potential positions are determined from the placement constraints defined for each labeling style. If no conflict is possible, then the label is fixed in its

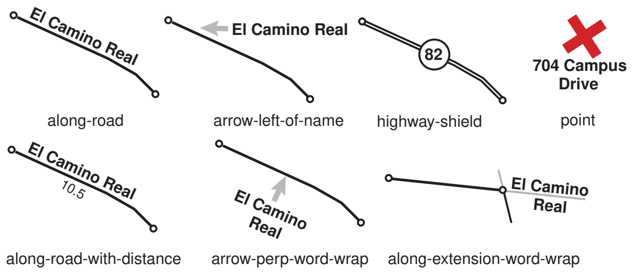

Figure 9: Several different labeling styles that might be used to label roads or landmarks along the route.

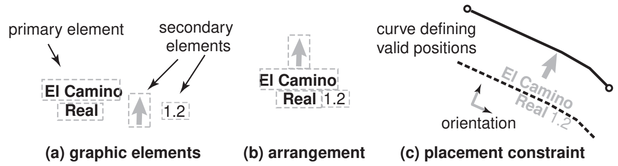

Figure 10: Components of a *labeling style*. (a) a set of graphic elements, (b) an arrangement of those graphic elements relative to a primary graphic, and (c) a constraint specifying the valid positions and orientations for the center of the primary graphic.

initial position and only those labels that are not fixed in this phase are placed during the label layout search.

The perturb function for the label layout search randomly picks a label to alter, randomly selects a labeling style for that label, and then randomly chooses a new location for the label from within one of the style's placement constraints. The label layout scoring function evaluates each label on the following criteria: (1) whether the label intersects or overlaps any other object in the map, (2) the proximity of the label to the center of its target, and (3) the rank of the chosen label style. The score for the complete map labeling is computed as the sum of the scores for each label.

Our general approach to the label layout problem is based on previous work on labeling point and line features in traditional geographic maps. Marks and Shieber [19] have shown that finding optimal label placements is NP-complete and several previous systems have used randomized search to find near-optimal label placements [29, 9]. These systems usually consider only a discrete set of possible locations and a single style for each label. LineDrive extends the search-based approach to handle a continuous range of label locations and a wider variety of potential labeling styles.

3.5 Context Layout

Although context features are secondary information not necessary for communicating the basic structure of route, they can improve the usability of a route map. LineDrive handles two forms of context: (1) linear features that intersect the main route, such as cross-streets, and (2) point landmarks along the route such as buildings and highway exit signs. We use the same basic approach for placing both cross-street and local landmarks. For brevity, we will describe the approach in terms of placing cross-streets[2].

Each cross-street is specified to the layout system by a piecewise linear curve of latitude/longitude points, the name of the cross street and an importance value for the cross-street. If the importance value is not pre-specified, we place highest importance on the last major cross-street just before each turning point. We have found that these streets are helpful as a warning that the turn is approaching. We initially compute the intersection point between every cross-street and the main route and then place the cross-streets at these intersection points. We also create a constraint region around the intersection

[2]Interested readers should consult [1] for the details of landmark layout.

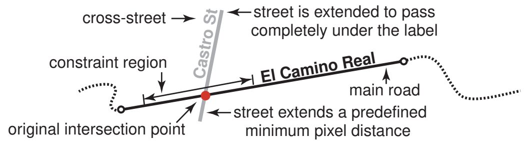

Figure 11: Placing a cross-street. The search considers placing Castro street within the constraint region as close to the original intersection point as possible. Once the cross-street and its label are placed, the cross street is extended to a minimum pixel length on either side of its base road and then further extended to pass under its label.

point which specifies the acceptable range of positions for the intersection point, as shown in figure 11. Cross-street labels are created just like main road labels and initially placed using the same rules.

The perturb function for cross-street layout randomly selects a cross-street and then randomly changes either the position of the intersection point between the cross-street and the main road, the position of the cross-street label, or whether the cross-street is hidden. Once the street is perturbed, we set the length of the cross-street to a predefined minimum extension length. Then, if the label has been placed directly above or below the street, we extend the street to pass completely over or under its label.

We score each cross-street based on four criteria: (1) the distance between the current position of the cross-street intersection point and the true intersection position, (2) the number of other objects the cross-street intersects, (3) the layout score of the cross-street label, which is computed using the same scoring function as for regular road labels, and (4) if the cross-street is hidden, we penalize the layout by an amount proportional to the cross-street importance.

Once the search phase of cross-street layout is complete, we clean up the layout. If the label of a cross-street overlaps any other object on the map, we remove the cross-street from the map. Label-object overlap can make the label difficult to read and obscure important route information. Since cross-streets are secondary features, removing them from the map is preferable to allowing such overlap. We do, however, allow the cross-streets to intersect other map objects. This is acceptable because cross-streets are thin, 1D objects, and are drawn underneath the other map objects in a light gray color so that they do not interfere with the legibility of the other objects. Finally, we clip each cross-street to every other road and cross-street in the route. This ensures that we do not introduce any false cross-street intersections in the maps.

3.6 Decoration

The decoration stage is responsible for adding four types of graphic decorations to the map to enhance its usability. Extensions on the ends of each road accentuate the turning points and help associate the road's label with the road. An orientation arrow shows the overall route orientation with respect to global north and can make it easier for navigators to geographically place the route. Bullets at each turning point show exactly where each turn decision must be made and help differentiate between roads that are headed in the same general direction. Finally, the rendering style for each road is set according to the type of the road.

Before adding extensions, we look up the pair of roads at each turning point in the database to check if they continue beyond the turning point. If a road does extend, we set the length of the extension to a predefined minimum extension length. If during label layout, the center of the road's label was placed directly above or below an extension, we grow the extension so that it passes completely over or under the label. Growing the extension in this manner helps form the proper association between the label and its target road. Finally, we clip the extension to all other roads and cross-streets.

To place the orientation arrow, we search the map image for an empty region large enough to hold the arrow. We accelerate the search by building a fixed resolution occupancy grid over the map image and only searching in empty cells of this grid. The search is

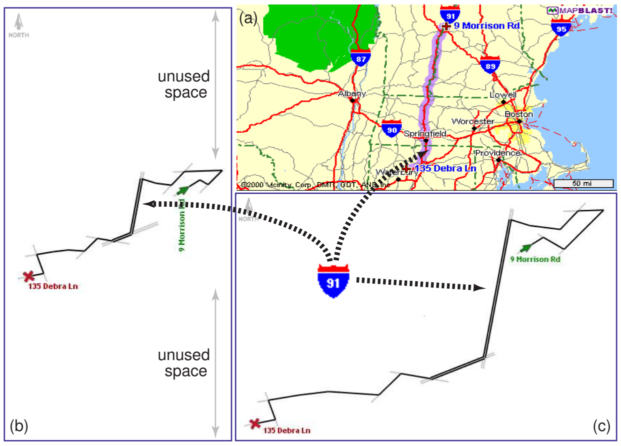

Figure 12: Selecting image size. (a) The route contains one long north-south road (I-91) and many short east-west roads near its origin and destination. (b) If the image size is selected based on the original north-south aspect ratio, the image is given more vertical space than horizontal space. The top and bottom of the image go unused because after growing all the short roads the aspect ratio of the map becomes much wider. (c) Computing the aspect ratio for selecting image size after growing the roads yields a horizontal image size and a more effective use of the space.

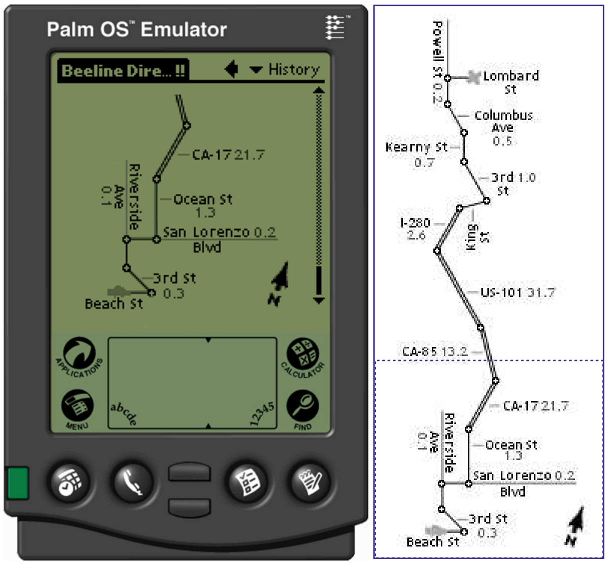

Figure 13: LineDrive map on a PDA. The route is rotated so that it fits the horizontally constrained image size of the PDA. The vertical dimension is unconstrained and users can scroll up to see the remainder of the route. This is the same route shown in figure 2.

ordered to first look for space in the four corners of the image and then search through the remaining image.

Our road database differentiates between three types of roads: limited access highways, highway ramps, and standard residential roads. In the decoration stage we set the rendering style for each road based on its type. Limited access highways are drawn as double lines, while ramps are drawn at half the thickness of the standard roads.

3.7 Image Size Selection

Since LineDrive designs route maps to fit within a given image size, the image size can have a large effect on the layout of the map. Consider a route map created for a predominantly north-south route that is designed to fit a wide aspect ratio viewport. All of the north-south roads would end up squashed while large regions of the image to the left and right of route would remain unused.

A better approach is to choose the viewport size based on the aspect ratio of the route. However, simply using the aspect ratio of the original uniformly scaled route does not always produce the desired result. Suppose, as in figure 12, the original route contains many east-west roads near its origin and destination, with one extremely long north-south road in between. Although the original aspect ratio for the route is north-south, after growing the short roads in our road layout, the aspect ratio of the route changes substantially. To estimate the aspect ratio of our final map before performing road layout, we initially fit all the roads at their original lengths to a large square viewport. We then grow all the short roads to their minimum pixel length and finally compute the aspect ratio of this new map, thus generating a more realistic estimate.

The image size of our maps may be limited by the resolution of the output device. Personal digital assistants (PDAs) usually have small screens, and long routes containing more than a few steps usually will not fit on these screens, even using our layout techniques. One solution is to split such routes into multiple segments, each containing a fixed number of turning points. The main drawback is that this approach requires flipping through multiple maps.

Another solution is to create a larger map image that can be scrolled. However, most PDAs provide good controls for scrolling vertically but not horizontally. In such situations, our image size is constrained only in the horizontal direction. Luckily, most routes have some predominant orientation. We find the predominant orientation by fitting a tight, oriented bounding box [11] to the route after growing all the short roads just as we did for the aspect ratio computation. We then fit the map to our horizontally constrained image by rotating the entire route so that the largest extent of the map is aligned with the vertical axis of the page. This approach provides extra space in the direction the route needs it most. As shown in figure 13, the orientation arrow helps indicate that the map has been rotated.

A common cartographic convention is that the north orientation arrow should align as closely as possible with the vertical axis of the page. Thus, we choose the rotation angle, either clockwise or counter-clockwise, which ensures that north arrow points in the upward semi-circle of directions. The rotation angle is bounded between ± 90.0 degrees and although the north arrow may not be aligned with the vertical axis of the page after the rotation it usually has a strong component in the vertical direction. Once the map has been verticalized, we can compute a vertical resolution for the image based on the number of steps in the route. We have empirically found that providing a vertical resolution of 200 pixels for maps with less than 10 steps, and adding 10 pixels for each step thereafter, works well.

4 Results

Examples of several route maps generated using LineDrive are presented in figure 14. We have tested the performance of the LineDrive system in two ways: (1) by collecting detailed statistics on a test suite of 7727 routes and (2) by providing web access to a beta version of LineDrive in order to receive user feedback.

Our test suite is comprised of 7727 routes queried over one day at `www.mapblast.com`. The median route distance for the test suite is 52.5 miles and the median number of turning points is 13. We ran each route through the system twice, first generating a webpage size image at a fixed resolution of 600 x 400 and then generating a PDA size image with a fixed horizontal resolution of 160 and a variable vertical resolution. The running time is largely dependent on the number of objects (i.e. roads, labels, etc.) that must be placed in the map. The median run time for a single map on an 800 MHz Pentium III was 0.7 seconds for the first run and 0.8 seconds for the second run. Although the vast majority of maps are clustered around these median times, a few outliers containing over 100 roads took about 13 seconds to generate for the webpage size.

A small percentage of the LineDrive maps generated from the

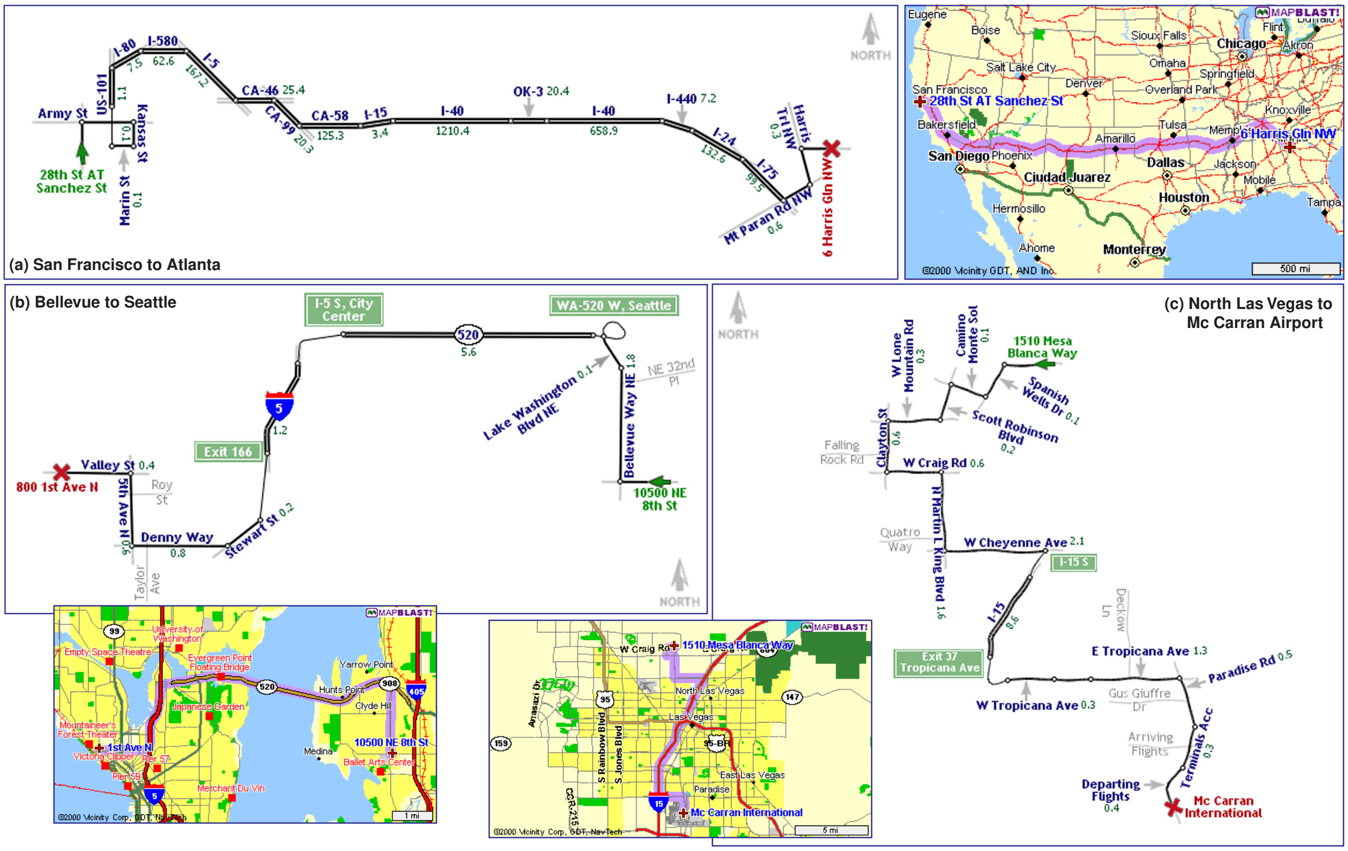

Figure 14: Examples of LineDrive maps with thumbnails of standard computer-generated maps for the same routes. (a) Non-uniform scaling allows all roads to be visible in this cross-country route. Since the ramp between Marin St. and US-101 intersects Army Street (actually passes above Army) it is not dropped from the map and proper intersection topology is maintained. (b) All ramps are maintained in this relatively short route from Bellevue to Seattle. Road shape is retained at both ends of I-5 in order to maintain a consistent turn angle with the adjacent ramps. The exit signs provide important context information for entering and exiting the highways. The highways are labeled using the *highway-shield* labeling style which helps differentiate the interstate, state and local highways from residential roads. (c) Cross-streets provide context and aid navigation in this route from North Las Vegas to McCarran Airport. The sketchy rendering style in this map is a subtle cue that the map is not drawn to scale.

test suite of routes contained layout problems such as topological errors or label-label overlap. In many cases, these problems were unavoidable because it is not always possible to make all roads large enough to be visible and simultaneously maintain the topology of the route. In a few cases, the problems could have been avoided but the randomized search did not converge to a near-optimal layout. The frequency of various layout problems for the 7727 route test suite are summarized in table 1.

The most significant problems that can arise in road layout are (1) that some roads may not be made large enough to be visible and clearly labeled or (2) that false or missing intersections may be introduced during the layout. Short roads, defined as less than 10 pixels in length, occurred in 5.3% of the webpage maps and 5.6% of the PDA maps. In most cases, the short roads could not be made longer either because there were a large number of roads all heading in the same direction or because lengthening the roads would have introduced a false or missing intersection. Although the PDA is horizontally constrained, the increase in the number of maps containing short roads is small because verticalization of these maps provides space for the short roads to grow. False and missing intersections occurred much less frequently than short roads and in all cases, avoiding the false or missing intersection would have required shrinking one or more roads to be extremely small.

The main problems that can occur in label layout are (1) that a label will be placed overlapping another label, or (2) that a label may be placed overlapping a road or landmark. Less than 0.5% of webpage sized maps contained overlapping labels, while 3.7% of PDA sized maps contained label-label overlap. This increase is due to the fact that long labels are especially difficult to place without overlap on the horizontally constrained PDA. Although label-road overlap occurs in a significantly larger number of maps, such intersections are much less detrimental to the overall usability of the map than label-label overlap.

The beta version of LineDrive was available to the public from October, 2000 until March, 2001 and served over 150,000 maps. Over 2200 users voluntarily filled out a feedback form describing their impressions of the LineDrive maps. While the group of respondents was self-selected, it is unclear whether any resulting bias would be positive or negative. Despite the potential bias, we believe that the feedback provides valuable insight into users' reactions to the maps. As shown in table 2, the general response to the LineDrive maps was overwhelmingly positive. Less than one percent of respondents said they would rather use the standard computer-generated maps than the LineDrive maps.

Nearly half of the respondents said they would like to use LineDrive maps in conjunction with standard maps. One difficulty with using LineDrive maps alone is that they provide little detail outside of the main route. If the navigator accidently strays from the route, it can be difficult to find a way back onto it. This can be especially problematic near the destination of the route where the navigator is less likely to be familiar with the area and may need to stray from the route in order to find parking. We address these problems on the website by providing a standard computer-generated map of the region near the destination of the route along with the LineDrive map.

Long distance trips often require more context than LineDrive

Performance Statistics	(7727 routes)			
	Web		PDA	
Median Time		0.7s		0.8s
Short Roads (< 10 pixels)	415	5.4%	430	5.6%
False Intersections	25	0.3%	23	0.3%
Missing Intersections	15	0.2%	14	0.2%
Label-Label Overlaps	37	0.5%	289	3.7%
Label-Road Intersections	901	11.7%	2096	27.1%

Table 1: Performance statistics for a test suite of 7727 routes with a median of 13 turning points per route and a median distance of 52.5 miles. Every row except for median time indicates the number of maps containing at least one instance of the problem. For example, the short roads row presents the number of maps containing at least one road less than 10 pixels long.

User Feedback		(2242 responses)
Would you use LineDrive maps in the future?		
1246	55.6%	Yes, I would use them instead of standard driving directions.
976	43.5%	Yes, I would use them along with standard driving directions.
20	0.9%	No thanks, I'll stick with standard driving directions.
How would you rate this feature?		
1787	79.7%	It's a blast.
253	11.3%	Just fine.
202	9.0%	Needs some work ...

Table 2: User feedback. The beta version of LineDrive has been accessed over 150,000 times and we have received 2242 responses to the system.

maps provide. While the cross-country map in figure 14(a) is a good stress-test showing that LineDrive can produce readable maps for routes containing many steps at vastly different scales, it is probably not the ideal map during such a long trip. Most navigators taking this trip would require a road atlas showing detailed local context along the way. LineDrive maps are designed for relatively short trips (i.e. under 100 miles) within a familiar region. Our experience is that most car-based trips fall within this range and the majority of people who use web-based mapping services generate directions to locations within their own greater metropolitan area.

About 9% of the respondents said the LineDrive system needs some work. However, most concerns were not with the LineDrive map, but instead with the particular route chosen by the route finding service. The beta version of LineDrive did not support cross-streets and local landmarks and the most common feature requests applicable to the maps were for the addition of cross-streets and exit signs. Based on the results of the beta test, LineDrive became the default map style for driving directions at `www.mapblast.com` in March 2001. This version supports cross-streets.

We have experimented with rendering LineDrive maps using a stroke-based, pen-and-ink style [18]. As shown in figure 14(c), the variations in the lines makes the map look more like a sketch than a precise computer-generated image. Strothotte et al. [24] have shown that rendering style can influence how people interpret architectural drawings, and we believe a similar principle applies to route maps. The sketchy rendering style is a subtle cue that the map is not drawn to scale.

5 Conclusions and Future Work

In this paper we have described a set of generalization techniques based on detailed study of the distortions made in handdrawn maps and designed to improve route map usability. We have also presented LineDrive, an automatic system for designing and rendering route maps that uses these techniques to ensure that all information required to follow a route is communicated clearly and concisely.

There are several directions for future research. We are currently exploring the use of insets as an approach for depicting route detail at turning points. The algorithm must automatically select the set of roads that should appear in each inset and the placement of the inset in the overall map.

Area landmarks, such as cities, and bodies of water, can make it easier for navigators to orient the route with respect to local geography. However, placing such landmarks in our maps can be difficult. In order to appear in their correct location with respect to the roads on the route, the size and shape of the area landmarks may need to be distorted. We are considering an approach that uses feature-based morphing [3] to incorporate such landmarks.

Acknowledgments: We are indebted to Christian Chabot for his invaluable help in designing, and championing LineDrive. Thanks to Vicinity corporation for allowing us to integrate LineDrive with their products and their help in extending LineDrive functionality. We are extremely grateful to Pat Hanrahan for giving us the freedom to pursue this line of research.

References

[1] M. Agrawala. *Visualizing Route Maps*. PhD thesis, Stanford University, 2001.

[2] T. Barkowsky, L. J. Latecki, and K. Richter. Schematizing maps: Simplification of geographic shape by discrete curve evolution. In C. Habel C. Freska, W. Brauer and K.F. Wender, editors, *Spatial Cognition II*, pages 41–53. Springer-Verlag, 2000.

[3] T. Beier and S. Neely. Feature-based image metamorphosis. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):35–42, July 1992.

[4] B. P. Buttenfield and R. B. McMaster, editors. *Map Generalization: Making rules for knowledge representation*. Longman Scientific, 1991.

[5] M. de Berg, M. van Krevald, M. Overmars, and O. Schwarzkopf, editors. *Computational Geometry: Algorithms and Applications*. Springer, 1997.

[6] M. Denis. The description of routes: A cognitive approach to the production of spatial discourse. *Cahiers de Psychologie Cognitive*, 16(4):409–458, 1997.

[7] M. Denis, F. Pazzaglia, C. Cornoldi, and L. Bertolo. Spatial discourse and navigation: An analysis of route directions in the city of Venice. *Applied Cognitive Psychology*, 13(2):145–174, 1999.

[8] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[9] S. Edmondson, J. Christensen, J. Marks, and S. Shieber. A general cartographic labeling algorithm. *Cartographica*, 33(4):12–23, 1997.

[10] G. Farin. *Curves and Surfaces for Computer-Aided Geometric Design*. Academic Press Ltd., 1988.

[11] S. Gottschalk, M. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Proceedings of SIGGRAPH 96*, pages 171–180, August 1996.

[12] J. Hershberger and J. Snoeyink. Speeding up the Douglas-Peucker line-simplification algorithm. In *5th Intl. Symp. on Spatial Data Handling*, pages 134–143, 1992.

[13] N. Holmes. *The Best in Diagrammatic Graphics*. Quarto Publishing, 1993.

[14] E. Imhof. *Cartographic Relief Presentation*. Berlin: de Gruyter, 1982.

[15] V. L. Interrante. Illustrating surface shape in volume data via principal direction-driven 3d line integral convolution. *Proceedings of SIGGRAPH 97*, pages 109–116, August 1997.

[16] A. M. MacEachren. *How Maps Work*. The Guilford Press, 1995.

[17] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.

[18] L. Markosian, M. A. Kowalski, S. J. Trychin, L. D. Bourdev, D. Goldstein, and J. F. Hughes. Real-time nonphotorealistic rendering. *Proceedings of SIGGRAPH 97*, pages 415–420, August 1997.

[19] J. Marks and S. Shieber. The computational complexity of cartographic label placement. Technical Report ITR-05-91, Center for Research in Computing Techniology, Harvard University, March 1991.

[20] Z. Michalewicz and D. B. Fogel, editors. *How to Solve It: Modern Hueristics*. Springer, 2000.

[21] M. Monmonier. *Mapping It Out*. The University of Chicago Press, 1995.

[22] U. Ramer. An iterative apprach for polygonal approximation of planar closed curves. *Computer Graphics and Image Processing*, 1:244–256, 1972.

[23] D. D. Seligmann and S. Feiner. Automated generation of intent-based 3D illustrations. *Computer Graphics (Proceedings of SIGGRAPH 91)*, 25(4):123–132, July 1991.

[24] T. Strothotte, B. Preim, A. Raab, J. Schumann, and D. R. Forsey. How to render frames and influence people. *Computer Graphics Forum*, 13(3):455–466, 1994.

[25] E. Tufte. *Envisioning Information*. Conneticut: Graphics Press, 1990.

[26] B. Tversky. Distortions in memory for maps. *Cognitive Psychology*, 13(3):407–433, 1981.

[27] B. Tversky. Distortions in cognitive maps. *Geoforum*, 23(2):131–138, 1992.

[28] B. Tversky and P. Lee. Pictorial and verbal tools for conveying routes. In C. Freska and D. M. Mark, editors, *COSIT*, pages 51–64, 1999.

[29] S. Zoraster. Practical results using simulated annealing for point feature label placement. *Cartography and GIS*, 24(4):228–238, 1997.